PUB-NO: EP000019515A2

DOCUMENT-IDENTIFIER: **EP 19515 A2**

TITLE: Electronic **postage meter** having improved **security** and fault tolerance features.

PUBN-DATE: November 26, 1980

INVENTOR-INFORMATION:

NAME	COUNTRY
QUATSE, JESSE TORRANCE	N/A
DODGE, DONALD EVERETT JR	N/A
DOVE, RICHARD KINGSLEY	N/A

ASSIGNEE-INFORMATION:

NAME	COUNTRY
FRIDEN CORP	US

APPL-NO: EP80400603

APPL-DATE: May 5, 1980

PRIORITY-DATA: US03757879A (May 9, 1979)

INT-CL (IPC): G07B017/04, G06F001/00 , G11C011/00 , G06F011/00

EUR-CL (EPC): G07B017/00 ; G06F011/14

US-CL-CURRENT: 235/101, 705/415 , 705/FOR.105

ABSTRACT:

CHG DATE=19990617 STATUS=O> A microcomputerized **postage meter** (5) that provides high degrees of **security** and fault tolerance. The meter maintains data **security** under low power conditions by the use of functionally nonvolatile memory units. Register and other data which must survive normal and abnormal losses of power to the meter electronics are stored in dual redundant battery augmented memories (hereinafter designated BAMs). Upon detecting an error condition, the microcomputer writes an appropriate fault code to the BAMs. A mechanism for disabling the meter includes dual redundant flip-flops which are set to a "faulted" state upon detection by the microcomputer of a failure condition. These flip-flops are powered by the BAM batteries. They cannot be reset except by physical access to the meter interior, which access is only available to authorized personnel at the factory. The fault flip-flops are also set when the microcomputer fails to properly execute its own operating program. Once the meter has been set to a "faulted" state, the fault flip-flops hold two signals, MPCLR and SYSCLR, true. The BAM contents may still be read out independently of the microcomputer which is prevented from accessing the BAMs. This is accomplished by allowing power necessary to read the BAMs to be supplied to the BAMs without supplying power to the microcomputer. Moreover, even if the microcomputer is powered, MPCLR prevents it from executing instructions and SYSCLR isolates it. The various register

values and the fault code then allow a reconstruction of the proper register values.

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: 0 019 515
A2

(12) EUROPEAN PATENT APPLICATION

(21) Application number: 80400603.9

(22) Date of filing: 05.05.80

(51) Int. Cl.³: G 07 B 17/04
G 06 F 1/00, G 11 C 11/00
G 06 F 11/00

(30) Priority: 09.05.79 US 37578

(43) Date of publication of application: 26.11.80 Bulletin 80/24

(84) Designated Contracting States: DE FR GB

(71) Applicant: FRIDEN MAILING EQUIPMENT CORPORATION
31285 San Clemente
Hayward California 94544(US)

(72) Inventor: Quatse, Jesse Torrance
37, Willow Lane
Sausalito California 94965(US)

(72) Inventor: Dodge, Donald Everett, Jr.
65 Buena Vista Terrace
San Francisco California 94117(US)

(72) Inventor: Dove, Richard Kingsley
180 Maxwelton Road
Piedmont California 94618(US)

(74) Representative: Weinmiller, Jürgen et al,
Zeppelinstrasse 63
D-8000 München 80(DE)

(54) **Electronic postage meter having improved security and fault tolerance features.**

(57) A microcomputerized postage meter (5) that provides high degrees of security and fault tolerance. The meter maintains data security under low power conditions by the use of functionally nonvolatile memory units. Register and other data which must survive normal and abnormal losses of power to the meter electronics are stored in dual redundant battery augmented memories (hereinafter designated BAMs). Upon detecting an error condition, the microcomputer writes an appropriate fault code to the BAMs. A mechanism for disabling the meter includes dual redundant flip-flops which are set to a "faulted" state upon detection by the microcomputer of a failure condition. These flip-flops are powered by the BAM batteries. They cannot be reset except by physical access to the meter interior, which access is only available to authorized personnel at the factory. The fault flip-flops are also set when the microcomputer fails to properly execute its own operating program. Once the meter has been set to a "faulted" state, the fault flip-flops hold two signals, MPCLR and SYSCLR, true. The BAM contents may still be read out independently of the microcomputer which is prevented from accessing the BAMs. This is accomplished by allowing power necessary to read the BAMs to be supplied to the BAMs without supplying power to the microcomputer. Moreover, even if the microcomputer is powered, MPCLR prevents it from executing instructions and SYSCLR isolates it. The various register values and the fault code then allow a reconstruction of the proper register values.

EP 0 019 515 A2

./...

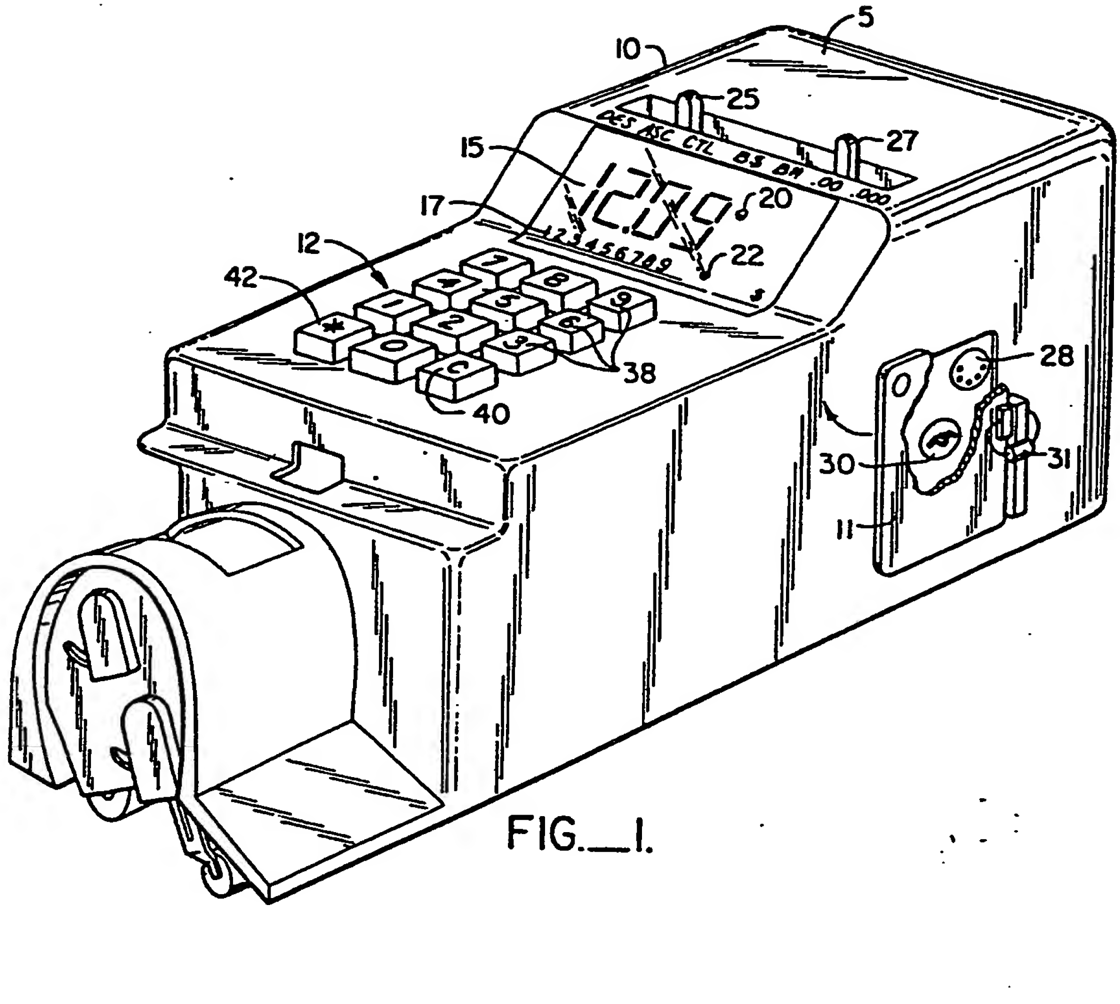

5
10
25
27
DES ASC CTL BS BM .00 .000
15
12.09
20
17
123456789
22
12
42
38
40
28
30
31
11

FIG._1.

6969-3

ELECTRONIC POSTAGE METER HAVING IMPROVED SECURITY AND FAULT TOLERANCE FEATURES

FIELD OF THE INVENTION

The present invention relates to a microcomputerized postage meter.

BACKGROUND OF THE INVENTION

Postage meters (hereinafter sometimes designated simply as "meters") are well-known devices for imprinting postage impressions of desired value either on a gummed tape or directly on an article to be mailed, thereby obviating the need to use postage stamps. Due to their convenience and flexibility, meters have found widespread use in commerce.

A postage meter normally includes a postage selection mechanism, a postage printing mechanism, and a plurality of internal registers for maintaining accounting information. The internal registers most commonly contain numerical values representative of the total postage paid for (control total), the total postage printed (ascending balance or ascending register), and the total postage remaining (descending balance or descending register). The information contained in the internal registers is redundant, since the ascending balance and descending balance normally sum to the control total.

Prior to using the meter a user must buy from a postal service employee a fixed amount of postage. (In this connection, the term "postal service" may refer to either a public or private mail carrying entity.) The postal service employee alters the contents of the internal registers to reflect the amount of postage paid by increasing the control

total and the descending balance by this amount. To use the meter, the user first selects a value of postage to be printed, and then activates the printing mechanism. The meter may be used until the descending balance reaches a predetermined minimum (e.g., until the postage paid for has been exhausted or has reached a minimum threshold value).

It can immediately be seen that postage meters are subject to stringent security requirements to ensure that all postage actually printed has been paid for. Thus, the level of security can be measured by the difficulty of activating the meter's printing mechanism without correspondingly updating the accounting registers within the meter, and also by the difficulty of altering or losing the meter register values, whether intentionally, inadvertently, or accidentally. To this end, the print mechanism and the accounting registers are located within a secure housing, and access thereto is restricted to postal service employees.

Postage meters have traditionally been essentially mechanical devices whose mechanical design is relatively complicated due to the need to correlate operation of the postage selection mechanism, the postage printing mechanism, and the registers. In particular, the print mechanism must print a postage value corresponding to the value set by the user, and the appropriate internal registers must be changed by this amount. Moreover, the meter must be interlocked to disable the print mechanism when the descending balance reaches the predetermined minimum level, and to prevent more than a single printing impression from being made during a cycle of the printing mechanism. Mechanisms capable of performing these functions, of necessity, contain a large number of mechanical parts, and therefore require considerable periodic maintenance. While several decades of experience have resulted in the design and implementation of acceptably reliable mechanical postage meters, such devices have still tended to be expensive, heavy, bulky, and slow.

Recent advances in the electronic arts have suggested the desirability of replacing many of the mechanical components in a postage meter with electronic components.

Thus, it is known in the prior art to provide a first-generation electronic postage meter employing discrete logic components. Such a meter is shown in U.S. Patent No. 3,938,095 to Check, Jr. et al.

By their nature, electronic postage meters rely heavily on continuous electric power during operation. However, frequent power loss of either a momentary or prolonged nature is to be expected. While power loss is not a particularly significant event for mechanical meters, it poses two distinct threats to the security of electronic postage meters. First, power loss presents a threat to the integrity of the register data which is typically stored in electronic memory units, since most electronic memories are volatile devices (i.e., they require continuous electric power to maintain their contents). This is to be contrasted with mechanical registers which are inherently nonvolatile devices. Second, the various correlation and interlocking functions are performed by electronic logic components, the performance of which can become unpredictable during a low power condition. Since this could lead to improper updating of registers, and the like, there must be provided a reliable mechanism wherein the electronic circuitry inhibits meter functioning when a low power condition is sensed. Moreover, this inhibiting must occur before the power falls to a level at which the electronic circuitry becomes unreliable.

In addition to the security requirements discussed above, a second requirement of postage meters, called "fault tolerance", comes into play when mechanical registers and other functions are replaced by electronic components. Fault tolerance refers to the meter's ability to maintain security in view of individual component failure. A postage meter is likely to be used in a variety of settings that may subject the components to environmental rigors such as mechanical shock, stray electric fields, and wide temperature variations, any of which may cause an electronic, mechanical, or electro-mechanical component to fail.

It is apparent that the large number of components in the first-generation electronic postage meters employing

discrete logic elements (i.e., transistors, diodes, etc.) tends to render such meters insufficiently reliable for postal service approval. A further difficulty with electronic postage meters employing discrete logic components is that the features and capabilities of the meter cannot be altered easily once the meter is constructed. Thus, like their mechanical counterparts, such meters cannot readily be adapted to new applications.

In recognition of the above problems, first-generation electronic postage meters employing discrete logic have given way to second-generation electronic postage meters employing large scale integration microcomputer architecture. An example of such a second-generation stand-alone postage meter is disclosed in U.S. Patent No. 3,978,457 to Check, Jr. et al., employing a microcomputer system which monitors the printing and other functions of the meter, and which supervises and maintains the required accounting information. A microcomputerized postage meter contains a smaller number of components than its discrete component counterpart, and is therefore likely to have improved fault tolerance characteristics. The fault tolerance can be further enhanced by the capability possessed by such a meter of verifying its own functions. Nevertheless, fault tolerance remains a potentially vexing problem because the very components that are used to check for failure are themselves subject to failure.

In spite of the numerous potential advantages of electronic postage meters over their mechanical predecessors, and further in spite of the expanded capability of microcomputerized postage meters, efforts to design an electronic postage meter having sufficiently high levels of security and fault tolerance to obtain postal service approval have been generally unsuccessful to date.

SUMMARY OF THE INVENTION

The present invention is a microcomputerized postage meter that provides sufficiently high degrees of security and fault tolerance to make the meter suitable for postal service approval.

Broadly, the postage meter of the present invention includes a microcomputer system, a keyboard or other suitable input means for entering data and instructions into the microcomputer, a display unit for allowing the operator to determine the status of particular registers, and a postage printing mechanism. The printing mechanism and the electronic components are located within a secure housing, and unauthorized access is impossible without physical destruction of parts of the housing. The housing mounts on a conventional meter base which performs the various paper handling functions and contains the power supplies.

The meter maintains data security under low power conditions by the use of functionally nonvolatile memory units. Register and other data which must survive normal and abnormal losses of power to the meter electronics are stored in dual redundant battery augmented memories (hereinafter designated BAMs). The BAMs are also used to store historical information for maintenance purposes (e.g., total usage of print mechanism), information relating to particular features or options (e.g., fractional cents), batch information (number of pieces and dollar amounts), and failure information (discussed more fully below). The BAMs are mutually independent, and each BAM includes a low power CMOS memory and a long life (5 years or more) battery. While each BAM can be read independently of the microcomputer, writing of new data into each BAM can only occur under microcomputer program control.

The meter maintains the integrity of its functioning under low power conditions by generating and responding to two timed signals when a low power condition is sensed. A first signal, designated "SYSCLR" (system clear), has the effect of inhibiting all meter functions that could have an effect on the printing of postage or the register values in the BAMs. A second signal, designated "MPCLR" (microprocessor clear), inhibits execution by the microcomputer. The generation of these two signals in a particular time-ordered manner prevents spurious operation during power up and power down periods.

When the supply voltage supplied to the meter falls below a predetermined threshold deemed necessary to ensure continued reliable functioning of the electronic components, a portion of the circuitry initiates a graceful power down sequence. After a sufficient time to allow completion of any ongoing BAM register updates (about 20 milliseconds), a SYSCLR signal is generated, i.e., SYSCLR goes true, inhibiting writing to the BAMs, and disabling the print mechanism. Then, after a delay (about 1 millisecond), an MPCLR signal is generated, i.e., MPCLR goes true, and the microcomputer is disabled. A capacitor within the meter retains sufficient charge at a voltage above that required by the electronic components to ensure that the circuitry operates reliably during the power down sequence.

An analagous sequence occurs when power is first applied to the meter circuitry during a power up cycle. When the supply voltage to the meter electronics is non-zero but still below the predetermined threshold level, MPCLR and SYSCLR go true. After a sufficient interval after the time that the voltage has risen to the predetermined threshold value to ensure reliable microcomputer operation (about 50 milliseconds for the particular microcomputer used), MPCLR goes false, which allows the microcomputer to start executing an initialization routine. Then after a delay (about 2 milliseconds) SYSCLR goes false to permit normal meter functioning.

The basic operating philosophy of the postage meter is to maintain security under normal operating conditions by having the microcomputer supervise and verify the various meter functions.

Printing is strictly supervised by the microcomputer. The meter employs a print mechanism comprising a print head which includes a plurality of print wheels, and a plurality of stepper motors corresponding to the plurality of print wheels, each motor controlling the positioning (indexing) of an associated one of the plurality of print wheels. Each stepper motor has verification means for generating a digital code, preferably binary coded decimal (BCD) code,

corresponding to the print wheel position. Locking means, such as a solenoid, under exclusive control of the microcomputer maintains the print head locked up in the home position except when printing is to be carried out in accordance with instructions from the microcomputer. Before freeing the print head for printing, the microcomputer verifies that the print wheels are in the positions corresponding to the desired amount of postage to be printed.

Changes to the register values in the BAMs can only occur under control of the microcomputer. In regular use, the meter is in a condition designated "mail room" mode wherein the only instructions available to the operator for changing the register values stored in the BAMS are the instructions for initiating a postage printing cycle. The effect of the print cycle is to decrement the descending register and increment the ascending register by the amount of postage printed. Register updating occurs only upon the receipt of a signal from the base that a clutch in the motor that drives the print head had been pulled or an internal signal that the print head has left its home position. As discussed above, the print head is freed only after the print wheel positions have been verified. Only authorized personnel can cause the microcomputer to execute a series of instructions to increase the control total value and descending register value by an amount corresponding to an amount of postage that is being added to the meter. Such instructions can only be executed when a mode-changing switch is actuated to switch the meter to a condition designated "post office" mode. In order to actuate the mode-changing switch, a seal must be broken and the switch unlocked with a special key available only to authorized personnel. The meter is not mounted on a base in the "post office" mode since no printing is to occur but rather, the meter is provided with external power to operate the electronic control circuitry.

All updating of register values, whether at the initiation of the print cycle or at the time of purchase of additional postage, occurs according to a checked arithmetic algorithm. The basic arithmetic constraint is that the

ascending and descending registers sum to the control total. The mechanics of the checking and the updating differ somewhat, depending on whether the meter is in the "mail room" or "post office" mode. In both cases, an extra "temporary" copy of the ascending and descending register values and control total is stored in the BAMs, in addition to the redundant "permanent" values. A consistency check is made to ensure that the register values in one BAM agree with the corresponding values in the other BAM. In the "mail room" mode the temporary values are updated as each character is entered on the keyboard; in the "post office" mode updating of the temporary registers occurs only after the entire keyboard entry is complete. At each update of the temporary register values, the values are checked for adherence to the arithmetic constraint. Updating of the permanent registers occurs during the print cycle in the "mail room" mode and in response to a particular keyboard sequence in the "post office" mode. The updated permanent registers are then checked for agreement between BAMs and for adherence to the arithmetic constraint.

It is apparent that the use of redundant BAMs, the maintenance of temporary register values in the BAMs, and the manner of updating the BAM registers prevent the loss of correct register values in the event of many types of malfunction. For example, if the microcomputer were to cease execution while updating the permanent registers in the BAMs in either the "post office" or "mail room" mode, the temporary registers would still provide sufficient redundant information to permit retrieval of the correct register values.

However, once a malfunction has occurred, the meter no longer possesses the redundancy necessary to maintain security in the event of a further malfunction. Accordingly, there is provided a mechanism for disabling the meter once a first malfunction is detected. This results in a meter that possesses a level of fault tolerance wherein security is maintained in spite of a single malfunction. The meter is still susceptible to loss of the correct register values if two independent malfunctions occur in a particular comple-

mentary way (e.g., loss of power to one BAM and memory failure of the other BAM; simultaneous loss of power to both BAMs; compensating arithmetic errors in both of the BAMs). Such combinations of malfunctions are extremely rare since the individual malfunctions are basically independent occurrences that are themselves rare.

The meter disabling mechanism includes dual redundant flip-flops which are set to a "faulted" state upon detection by the microcomputer of a failure condition. These flip-flops are powered by the BAM batteries. The setting of these fault flip-flops lights an indicator lamp and generates a SYSCLR signal to inhibit all meter functions that could have an effect on the register values in the BAMs. In addition, the setting of the fault flip-flops generates an MPCLR signal to cause the microcomputer to cease execution. Setting of the fault flip-flops is inhibited by the presence of a SYSCLR signal, so that the fault flip-flops cannot be set during power up and power down cycling. The fault flip-flops, once set, hold SYSCLR and MPCLR true whenever power is supplied to the meter. They cannot be reset except by physical access to the meter interior, which access is only available to authorized personnel at the factory.

The fault flip-flops are also set when the microcomputer fails to properly execute its own operating program. Under proper conditions, the microcomputer periodically generates a signal which triggers two independent monostable multivibrator circuits, preferably retriggerable one-shots, to maintain a particular logic level on the one-shot output. The microcomputer responds to the detection of a fault condition by <u>not</u> triggering the one-shots. When there appears on the one-shot output a logic level that corresponds to the failure of the microcomputer to generate this signal, the fault flip-flops are set. In addition to programmed failures to trigger the one-shots, any circumstance (such as microcomputer failure) that prevents the microcomputer from triggering the one-shots has the effect of causing the meter to fault. Thus, the mechanism for setting the fault flip-flops exploits the microcomputer's ability to diagnose failures but may also be activated should the microcomputer itself fail.

Adherence of the register values to the arithmetic constraint, and consistency between the BAMs, as described above, are considered absolute prerequisites to continued meter operation. Thus if an arithmetic check yields inconsistent results, or if corresponding registers in the two BAMs disagree, the meter faults after writing an appropriate failure code to the BAMs. All writing to the BAMs is verified, and an error causes the meter to fault, preceded by the writing of an appropriate failure code.

During the printing operation, the microcomputer and associated logic elements test for various indications of malfunction of the print value setting and printing mechanisms. In particular, if during a printing operation the print head leaves its home position before the print wheels are positioned or the print head fails to leave its home position within a predetermined maximum time period (100 milliseconds) of the receipt of a clutch signal, the meter faults. In such cases, a numeric code representative of the particular problem detected is written to the BAMs prior to deactivation of the meter.

Other types of malfunction do not pose an immediate threat to security, and tend to be self-curing or one-time occurrences. These situations are handled by turning off the displays and stepper motors, locking the print solenoid, and causing the microcomputer to execute a trivial wait loop while still continuing to fire the one-shots. This condition is designated a "soft fault". The typical cause of a soft fault is some sort of mistake which necessitates suspending operation to prevent a second mistake which could cause an error. The mistake may arise from operator error, base malfunction, dirt, or noise. An example of soft fault is the apparent failure of a print wheel to move from one position to an adjacent position within a predetermined maximum time period (12.8 milliseconds). In many instances, the cause might be due to imperfect operation of the verification means, such as poor electrical contact due to dirt. In such a case, reinitializing the meter which causes the motors to be stepped through their entire range would cure this

problem. Another example of a soft fault is the detection of a clutch signal when it is not expected. This could be caused by a malfunction in the base or it could result from improper paper handling by the operator. Again, reinitializing the meter would cause this problem to go away. A further example of a soft fault is the inability to verify a BAM address. Since the BAM address lines communicate outside the meter housing, they are susceptible to electrical noise which could cause a bad address verification. Again, this problem is likely to be intermittent in nature and does not pose a threat to security so long as no attempt is made to write to the BAMs.

Yet a further potential threat to the meter's security is present if the mode-changing switch malfunctions, giving a spurious indication to the microcomputer that the meter is in the "post office" mode since this could allow an unauthorized (unpaid for) increase of the control total and descending registers. To prevent such an occurrence, the microcomputer checks the mode-changing switch during initialization and at the beginning of a print cycle. If the meter is found to be in the "post office" mode while apparently mounted on a base at either of these times, a soft fault condition is considered to have occurred. One possible cause of such a condition is if a postal service employee actuates the mode changing switch while the meter is mounted on a base. So long as operation is suspended, no threat to security exists, and the soft fault condition is cured by having the postal service employee reinitialize the meter under external power (meter off base) or on the base but in the "mail room" mode.

During initialization, the microcomputer checks to make sure that no nonzero failure codes have been written to the BAMs. Once an error code has been written to the BAMs and the meter has faulted, it should be impossible to initialize the meter since the fault flip-flops hold MPCLR true when power is supplied. Thus, detection of a nonzero failure code is a possible indication of a BAM failure or an inability to read a good zero from the BAMs, and causes the meter

to fault. Due to the nature of the error, no attempt is made to write a failure code.

Once the meter has been set to a "faulted" state, the BAM contents may still be read out independently of the microcomputer. During this time, the microcomputer is prevented from accessing the BAMs. This is accomplished by allowing power necessary to read the BAMs to be supplied to the BAMs without supplying power to the microcomputer. Moreover, even if the microcomputer is powered, MPCLR prevents it from executing instructions and SYSCLR isolates it. The various register values and the fault code then allow a reconstruction of the proper register values.

For a further understanding of the nature and advantages of the invention, reference should be had to the ensuing detailed description taken in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a perspective view of the exterior of a microcomputerized postage meter according to the present invention;

Fig. 2 is a perspective view of part of the print mechanism located within the housing of the meter;

Fig. 3 is a schematic block diagram of the electronic meter system;

Fig. 4 is a system flow chart illustrating the general operation of the invention;

Fig. 5 is a circuit diagram of the external power levels that are communicated from the base to the meter;

Fig. 6 is a circuit diagram of signals communicated from the base to the meter;

Fig. 7 is a circuit schematic of circuitry for generating certain internal signals within the meter;

Fig. 8 is a circuit schematic of the power surveillance and system reset circuitry of the present invention;

Fig. 9 is a circuit schematic of the circuitry that inhibits meter functioning upon detection of a fault condition;

Fig. 10 is a timing diagram illustrating the voltage levels responded to by the power surveillance circuitry of Fig. 8, and the signals generated by the system reset circuit of Fig. 8;

Figs 11a, 11b, and 11c, taken together form a circuit schematic illustrating the microcomputer and BAM control circuitry;

Fig. 12a and 12b, taken together, form a circuit schematic illustrating the microcomputer circuitry for refreshing the displays and reading the switches;

Fig. 13 is a circuit schematic illustrating circuitry for controlling the print wheel motors and reading the verification contacts;

Fig. 14 is a schematic illustrating a suitable memory allocation of registers required by the operating program of the microcomputer;

Fig. 15 is a schematic illustrating a suitable bit allocation for the switch registers;

Fig. 16 is a schematic illustrating a suitable bit allocation for the status and print routine registers;

Fig. 17 is a flow chart of the foreground routine;

Fig. 18 is a schematic illustrating the organization of the BAMs and the memory allocation of register values stored therein;

Fig. 19 illustrates in tabular form the temporary register contents after various operations;

Figs. 20a, 20b, and 20c, taken together, form a flow chart of the print task routine;

Figs. 21a and 21b, taken together, form a flow chart of the keyboard task routine.

DESCRIPTION OF THE PREFERRED EMBODIMENT

Turning now to the drawings, Fig. 1 is a perspective view of the exterior of a postage meter 5 according to the present invention. Meter 5 includes an exterior housing 10 for containing a print mechanism and electronic control system described in detail below. In operation, meter 5 is installed on a mailing machine base, not shown, which contains power supplies for the meter, performs paper handling

functions (envelopes or tapes), communicates synchronizing signals to the electronic control system, and provides mechanical power for activating (but not setting) the print mechanism. Since the mailing machine base comprises a conventional ancillary device, a detailed description of the elements thereof is omitted to avoid prolixity.

The exterior components of meter 5 generally include a control panel and a door behind which is a postal service employee's panel. The control panel includes a keyboard 12, displays 15 and 17, indicator lights 20 and 22, and selector switches 25 and 27. The postal service employee's panel includes a remote power connector 28 for powering the meter when it is removed from a base, and a keylock operated mode changing switch 30 which, when actuated, places the meter in a "post office" mode to permit the postal service employee to change the register values to reflect a purchase of additional postage. Access to connector 28 and switch 30 requires the breaking of a seal 31 and subsequent sliding of door 11.

The plurality of keyboard keys, switches 25, 27, and 30, and displays 15 and 17 allow the operator or postal service employee to communicate with the internal electronic control system and to specify the necessary control signals for operation of the electronic control system and the print mechanism. Meter 5 may also operate under control of external devices (e.g. an electronic weighing station) and thus includes suitable connections for communicating electrical signals to the electronic control system within housing 10. Hereinafter, the term "switches", when used generally, will be taken to include the above switches and keys, the base synchronization signals, and signals from external devices.

Keyboard 12 comprises a calculator configured digit field including ten numbered keys 38, and special function keys 40 and 42, designated clear KB and clear batch, respectively. Data entry is effected by manually actuating numbered keys 38 in a sequence corresponding to the desired entry. The most significant digit is entered first, normally with a maximum dollar entry capacity of four digits (99.99 or

9.999 if a fractional cents meter). Attempts to continue entry beyond the four digit maximum are ignored. This entry capacity expands to up to eight digits when the meter is in the "post office" mode, the maximum number of digits being a preset parameter.

Display 15 is preferably a four-digit numeric display having characters sufficiently large to be readily visible in normal operation. A character height of about 0.6 inches is suitable. As data is entered via keys 38, it is displayed on display 15. As values are keyed in, they enter the right-most digit and are shifted left for each valid subsequent keyboard entry. Depression of clear KB key 40 clears the keyboard entry and display 15. Simultaneous depression of clear batch key 42 and clear KB key 40 initializes internal batch count and batch dollars registers. Display 17 is preferably a nine-digit numeric display and is used for displaying register values. Since this display is generally used less often than display 15, a smaller character size (e.g., approximately 0.125 inches) than that used for display 15 is suitable. Five position slide switch 25 allows the user to select an internal register to be displayed on display 17. In particular, descending register value, ascending register value, control total value, batch dollar amount, and batch count may be displayed. Two-position selector switch 27 has effect only if meter 5 has been pre-programmed as a "fractional cents" meter. In effect, switch 27 allows the operator to overide the fractional cents aspect during keyboard entry so that the last digit entered corresponds to cents rather than tenths of a cent.

When lit, indicator lights 20 and 22 serve to warn the operator that a condition has occurred which makes it necessary to at least temporarily suspend operation of the meter. Indicator light 20, designated "Service", is lit when the meter has been internally set to a faulted state. When light 20 is lit, the meter is inoperative, and must be returned to the factory before it can be used again. In addition, no set of keyboard entries or sequence of powering

the meter can restore operation, and access to the housing interior, possible only at the factory, is required to place the meter in operative condition. Indicator light 22, designated "Add $", is lit when the amount of postage entered would cause the descending register value to become negative upon printing. Depressing "clear entry" key 40 extinguishes light 22 and allows subsequent meter operation at postage values which will not cause the descending register value to become negative.

The amount of postage available for printing may be changed by a postal service employee by placing meter 5 in the "post office mode". This is possible only when the postal service employee breaks seal 31 on switch 30, and enables movement of switch 30 to the "post office mode" position by actuation of keylock 32. When this occurs, keys 40 and 42 and switch 25 take on different functions. When switch 25 is moved to either of the batch register display positions, nine-digit display 17 functions as a keyboard verification display. Keyboard keys 38 retain their normal function of data entry, but special keys 40 and 42 allow the postal employee to change the values of the accounting registers. In particular, depression and release of clear batch key 42 causes the keyboard entry to be added to the descending register and the control total values. If clear KB key 40 is depressed while clear batch key 42 is also depressed, the keyboard entry is subtracted from the descending register and control total values when "clear batch" key 42 is released.

Fig. 2 is a perspective view of portions of the print mechanism located within housing 10 of meter 5. The print mechanism comprises a print head having a plurality of print wheels, a plurality of stepping motors (hereinafter sometimes referred to as steppers) to independently set the print wheels, and means for moving the entire print head to a printing position without disturbing the print wheel settings. Fig. 2 shows the print wheel setting means for a single digit. A print wheel 43 having a plurality of value indicating print surfaces 44 is driven by a stepping motor

45. Additional motors 46, 47, and 48, while not shown in Fig. 2, are shown schematically in Figs. 3 and 13. Stepping motor 45 drives a gear 49 which engages a rack 50. Rack 50 is rigidly connected to a drive yoke 52 which carries a rotatably mounted drive ring 55. Drive ring 55 is slidably mounted on a rotatable bar 57 which has a channel 60 along its length. A rod 62 is disposed within channel 60, being coupled to drive ring 55 at one end and terminating in a rack portion 65 at the other end. Print wheel 43 has a gear 68, and rack 65 drives print wheel 43 through gear 68 and an idler gear 70.

Thus, rotation of gear 49 on the output shaft of motor 45 causes movement of rack 50 along the direction of bar 57. Movement of drive yoke 52 is transmitted through rod 62 and idler gear 70 to gear 68 and print wheel 43. Motor 45 is coupled to a position sensor 75, which in the preferred embodiment comprises verification contacts for generating a binary code representative of the position of an internal wiper connected to the motor shaft. Additional position sensors 76, 77 and 78 are associated with motors 46, 47, and 48, respectively, and while not shown in Fig. 2, are shown schematically in Figs. 3 and 13.

It should be understood that a separate subassembly comprising a stepping motor, a motor gear rack, a position sensor, a drive yoke, a drive ring, a rod, a print wheel rack, an idler gear, and a print wheel substantially similar to elements 45, 50, 75, 52, 55, 62, 65, 70, and 43 is provided for each digit that can be printed. Thus, in the preferred embodiment where four digits may be printed, there are four such subassemblies. However, there is only one channeled bar 57, channel 60 being sized to accomodate four rods (including rod 62) in a side by side configuration.

Printing occurs by a rotation of bar 57, the mechanical power being supplied by the base. It should be noted that the drive yokes, while movable along bar 57, are prevented from rotating. Rather, bar 57 rotates independently. The print head (which includes the print wheels) is maintained in a fixed relationship with respect to bar 57 so that

rotation of bar 57 brings the print head from a normally upwardly facing disposition into a downwardly facing position for printing, during which the print wheels are exposed at an opening in the bottom of housing 10. A solenoid 85, shown schematically in Fig. 12 prevents movement of the print head in the absence of a specific print enabling signal.

Fig. 3 is an electrical schematic showing in functional block diagram form the electronic circuitry located within meter 5. Whenever meaningful, reference numerals corresponding to those associated with the physical representation of the same components in Figs. 1 and 2 are used.

The central element in the meter electronic circuitry is a microcomputer 90 having contained program memory, working memory, and processing capabilities. During operation, microcomputer 90 is responsible for performing several functions, including periodic refreshing of the displays, periodic sampling of all the switches, communication with external devices, detection of synchronizing signals from the base that signify the initiation of a print cycle, initiating and verifying print wheel setting, and performing accounting functions. Since microcomputer 90 is required to communicate with a relatively large number of peripheral devices, it operates in conjunction with input/output (hereinafter designated I/O) expansion circuitry. The I/O expansion circuitry includes I/O expanders 92 and 93, demultiplexers 94 and 95, and multiplexers 96 and 97. As will be described in detail below, I/O expander 92 provides signal paths, via demultiplexers 94 and 95 and multiplexer 96, between microcomputer 90 and the displays and switches. The switch status is communicated to microcomputer 90 on an output line 98 of multiplexer 96. In a like fashion, I/O expander 93 allows microcomputer 90 to send signals for driving the stepping motors, while multiplexer 97 allows microcomputer 90 to receive information on a line 99 from the motor position sensors.

In addition to the switches, displays, motors, position sensors and I/O expansion circuitry described above, the circuitry includes dual redundant battery augmented

memories 100 and 102 (BAMs). The BAMs are used to store information which must survive normal and abnormal losses of electrical power to the meter. This includes register data (ascending register value, descending register value, control total value), batch count information, and failure information, all to be discussed in greater detail below.

The circuitry also includes power surveillance circuitry 105 for detecting what is anticipated to be a low power situation, and system reset circuitry 110 that responds to low power conditions by generating timed signals, designated "SYSCLR" (system clear) and "MPCLR" (microprocessor clear), in a particular time ordered manner for a graceful power up or power down sequence. Dual redundant fault flip-flops 111 and 112 are coupled to the system reset circuitry to inhibit meter functioning once the flip-flops are set to a "faulted" condition. Flip-flops 111 and 112 are maintained in a non-faulted state by the outputs of dual redundant retriggerable one-shots 113 and 114, respectively. During operation, so long as no failure condition is detected, microcomputer 90 periodically sends a signal to one-shots 113 and 114 to prevent flip-flops 111 and 112 from being set. Setting of flip-flops 111 and 113 occurs whenever the microcomputer fails to send a signal to one-shots 113 and 114 within a predetermined time period. This occurs either when the microcomputer detects any one of a number of failure indicating conditions, or when the microcomputer itself fails to properly execute its operating program.

Microcomputer 90 communicates with the remaining circuitry primarily via three eight-bit data buses 120, 125, and 130, designated DB, P1, and P2 respectively. Individual lines on buses 120, 125, and 130 are designated DB0-DB7, P10-P17, and P20-P27, respectively. Buses 120 and 125 communicate with external peripheral I/O devices, designated generally by reference numeral 131. I/O devices 131 communicate signals which cause microcomputer 90 to carry out the same tasks as it would in response to corresponding switch manipulations. Within meter 5, certain of these data lines serve dual functions. In particular, bus 125 communicates

with BAMS 100 and 102 for supplying an address, and with displays 15 and 17 through a buffer 128 on an eight-bit line 129 for supplying a digit code. Also, lines 132 and 133 of bus 130 serve the dual function of selecting which of BAMS 100 or 102 is to be accessed and of selecting which of I/O expanders 92 and 93 is to be accessed via four-bit line 134 of bus 130. Additionally, a four-bit line 135 from I/O expander 92 communicates with the BAMs for supplying data and further communicates with multiplexer 97 for selecting a particular bit from one of motor position indicators 75-78.

Having discussed the general structure of the meter electronic circuitry, the general operation of meter 5 under the control of microcomputer 90 may be described. Broadly, microcomputer 90 executes instructions according to an operating system which includes an initialization routine, a foreground routine, a power loss interrupt routine, a background dispatcher loop, and a plurality of background tasks, including subroutines. The background dispatcher loop is, in effect, the main program, and will sometimes be referred to as such. Fig. 4 is a flow chart illustrating the relationship between the initialization routine, the background dispatcher loop, and the background tasks.

Given the conflicts on buses 120 and 130 it is necessary that data or addresses on these lines that could affect access to the BAMs not be disturbed by an attempt to use one of the lines for its alternate function. Thus, the operating system of microcomputer 90 maintains a sharp division between those portions of the program that access the BAMs (designated background) and those portions of the program according to which microcomputer 90 communicates with the switches, displays, and the like (designated foreground).

The background dispatcher loop and the various background tasks are responsible for the normal meter functions including print supervision and accounting, and rely on information supplied by the foreground routine. The foreground routine is executed in response to a periodic timer interrupt generated by a timer inside microcomputer 90 and has the basic functions of energizing the displays, reading

the switches and base synchronization signals, and sending signals to one-shots 113 and 114 to prevent fault flip-flops 111 and 112 from faulting the meter. The state of the various switches ascertained by the foreground routine is stored in switch registers within the working memory of microcomputer 90, so as to provide information to other parts of the operating system. The basic constraints on the timer interval are imposed by the requirements that the display digits be energized a particular number of times per second and that the keyboard switches be read more often than the minimum duration of a switch closure. As will be seen below, the rate at which the one-shots must be triggered is an easily varied parameter.

Upon receipt of a signal from power surveillance circuitry 105, an interrupt is generated, and the power loss interrupt routine is entered. This routine blanks displays 15 and 17, deenergizes stepping motors 45, 46, 47, and 48, and sets solenoid 85 to the print disabling position. The routine also sets a bit in memory, in analogy to the switch registers to communicate to the background that a power loss interrupt has occurred.

Turning to the background, with reference to Fig. 4, after execution of the initialization routine (block 150), microcomputer 90 sets a bit in its memory to signify that no BAM accesses are in progress, this state being designated "enable foreground" (block 152), and enters the background dispatcher loop indicated generally within dashed rectangle 154. Within this loop, microcomputer 90 sequentially checks the status of the power loss bit and various so-called WAKEUP bits that will have been set by the foreground routine, and jumps to an appropriate background task as required. The first check is whether the power loss bit has been set (block 158). If it has, microcomputer 90 executes a wait loop 160 until the system reset circuitry generates an MPCLR signal to cause microcomputer 90 to cease execution. Assuming no power loss, microcomputer 90 sequentially checks the WAKEUP bits to determine whether a register display is required (logical branch 162), whether the meter is in a print cycle (logical

branch 164), whether an I/O request has been made (logical branch 166), whether the keyboard register has changed indicating a new keyboard entry (logical branch 168), and whether a print task is to be performed (logical branch 170), before returning to the beginning of the background dispatcher loop. During a print cycle, I/O requests and new keyboard entries are ignored, as indicated by branch 171.

Assuming that a register is to be displayed, that an I/O request is present, that a new keyboard entry has been made, or that a print task is to be performed, the program jumps to the appropriate background task indicated by blocks 172, 176, 178, and 180 respectively. The particular features of the background task routines will be described in detail below. Since execution of each of these background tasks involves access to the BAMs at some point, the foreground is immediately disabled so that data on the lines to the BAMs will not be overwritten during the execution of the foreground routine which occurs in response to a timer signal not under the control of the background dispatcher loop. After completion of the appropriate tasks, each of the background routines returns to the background wait loop after enabling the foreground (block 152). Under certain circumstances, the print task routine does not return directly to the main background loop, but first jumps to a portion of the keyboard task return before returning (blocks 182 and 183).

A detailed description of the foreground routine is deferred until after the structure of the meter electronic circuitry has been set forth in greater detail. Nevertheless, the following general description is set forth in order to make clear the overall operation of the meter operating system. The foreground routine does not accomplish all its tasks on a single entry. Rather, a total of 27 timer interrupts are required for a complete cycle. Thus a 400 microsecond timer interval results in a cycle time of about 11 milliseconds, which correlates with the need for the digits of display 17 to be energized approximately 80 times per second. Also, the 11 millisecond interval is considerably shorter than the 50 millisecond minimum duration of a pulse resulting from the depression of a keyboard key.

The digits on display 15, requiring greater brightness than those on display 17 must be energized more often. In particular, display 15 is energized four times for every time that on display 17 is energized. As will be discussed in greater detail with reference to Figs. 12a and 12b, sending a signal to energize a particular digit on display 15 or 17 allows a particular position of register display selector switch 25 or a particular keyboard switch 38 to be read. The state of the remaining switches must be separately determined. Since switch noise having a duration of approximately 2 milliseconds is inevitably present during the initial actuation of a switch, the foreground routine debounces the switches and does not consider a switch state to have changed unless it remains changed on two successive readings (i.e., for more than 11 milliseconds).

Upon entering the foreground routine in response to an interrupt from the 400 microsecond timer, microcomputer 90 checks the foreground bit to determine whether the foreground is enabled or whether a background task was in progress. If the foreground is enabled, the routine does the appropriate display energization and/or switch state read according to a sequence to be described below. Assuming the foreground is enabled, the routine then checks whether the interrupt is a particular one in the sequence that requires a signal to be sent to the fault one-shots and only triggers the one-shots at that particular point in the sequence. If the foreground was not enabled, the routine triggers the fault one-shots regardless of the interrupt's position in the sequence control and then passes to the background routine.

The discussion immediately following is with reference to Figs. 5 - 13 which are circuit schematics illustrating in greater detail the electronic meter system set forth in the block diagram of Fig. 3. Accordingly, in the discussion that follows, reference should, at appropriate times, also be made to Fig. 3. The circuit elements are, wherever practical, solid state integrated circuit components. Suitable components are set forth in the following table:

Function	Type	Manufacturer
Microcomputer	8049	Intel
BAM	P5101L-1	Intel
I/O Expander	P8243	Intel
Comparator	LM2903P	National Semiconductor
Buffer	ULN2003AN	Texas Instruments
One-shot	MC14538BP	Motorola
NAND gate	MC14011P	Motorola
NOR gate	MC14001	Motorola
OR gate	MC14071	Motorola
Multiplexer	74C150	National Semiconductor
Demultiplexer	74145	Texas Instruments

It should be noted that the preferred type of microcomputer (Intel microcontroller 8049) includes 2048 bytes of read only memory (hereinafter designated ROM) which is programmed with the desired operating program by the manufacturer. During the period that the invention was undergoing development, a type 8035 microcontroller having no ROM of its own was used in connection with an Intel 8755A memory chip which provides 2048 bytes of UV programmable ROM, to facilitate changes to the operating program.

Figs. 5 and 6 are circuit diagrams illustrating external power levels and signals that are communicated from the base to the postage meter. Communication is preferably established by a multiple conductor cable having suitable connectors at its ends to mate with corresponding sockets on the base and on the meter.

Turning first to Fig. 5, various power levels are shown. An external source of at least +9.6 volts, designated VLOGIC is coupled to a line 200 in meter 5. The voltage on line 200 charges a large (e.g. 10,000 microfarad) capacitor 201. Line 200 communicates to an input of a 5-volt regulator 202 to produce a regulated +5 volt power level, designated VCC, on a line 205. VCC is used for powering various integrated circuit components within meter 5 that require a regulated +5 volt source.

BAMs 100 and 102 are powered by separate voltage sources, designated VBAM1 and VBAM2. These voltages are

produced on respective lines 210 and 212, within meter 5. A related +5 volt level, designated VBAM0, is generated on a line 215. A first pair of 2.8 volt batteries 220 is coupled to line 210 via three diodes 225 in series. Similarly, a second pair of 2.8 volt batteries 230 is coupled to line 212 through three diodes 235 series. VLOGIC is coupled to a line 240 within meter 5, and is regulated to +5.7 volt by a 5 volt regulator 242 and a diode 245 to ground. The regulated output voltage appears at a circuit point 250 and is communicated to line 215 through a diode 251. Circuit point 250 is coupled to line 210 through a diode 252 and to line 212 through a diode 255.

Two levels, designated VEXT1 and VEXT2, are communicated to respective lines 258 and 260 within meter 5. However, these lines are not powered when the meter is on a base, but rather are powered independently and mutually exclusively of VLOGIC when the BAM contents are to be read independent of microcomputer 90. VEXT1 also communicates to lines 210 and 215 through respective diodes 262 and 263. Similarly, VEXT2 communicates to lines 212 and 215 through respective diodes 265 and 266.

Since the voltage from regulator 242 exceeds the voltage of batteries 220 and 230, when VLOGIC is supplied to line 240, the regulated output of regulator 242 supplies power on lines 210 and 212. This produces a level of +5 volt on lines 210 and 212, there being a voltage drop of approximately 0.7 volts across each of diodes 252 and 255. When there is no power from regulator 242, batteries 220 and 230 provide power to lines 210 and 212, respectively. Batteries 220 and 230 are preferably lithium primary cells having a shelf life typically in excess of ten years. Suitable batteries are manufactured by Mallory.

Two additional external power sources in the base are coupled to meter 5. A source of +34 volt, designated VMOTOR, is coupled to a line 270 within meter 5 to supply power to stepping motors 45, 46, 47, and 48. A source of +5 volt, designated VDISP, is coupled to a line 275 within meter 5 to supply power for the digits of displays 15 and 17.

In addition to supplying power to meter 5, the base performs paper handling functions, and communicates signals to meter 5 in order to synchronize the activity within meter 5 with the paper handling that is ongoing. These signals are illustrated schematically in Fig. 6. A grounded connection in the base is communicated to a line 280 within the base to supply a signal designated $\overline{\text{MB}}$, which is low whenever the meter is mounted to a base. A signal, designated $\overline{\text{TAPE}}$ is communicated to a line 282 within meter 5, a low level signifying to the meter that a postal tape is to be printed. A signal, designated $\overline{\text{ENVELOPE}}$ is communicated to a line 285 within meter 5, a low level signifying that an envelope is to be printed. A signal, designated $\overline{\text{REPEAT}}$, is communicated to a line 287 within meter 5, a low level signifying that multiple tapes are to be printed with the same value. A signal, designated $\overline{\text{CLUTCH}}$, is communicated to a line 290 within meter 5, a low level signifying the fact that a clutch within the motor in the base that provides mechanical power to the print head within the meter has been pulled. An external interrupt signal, designated $\overline{\text{EXTINT}}$, is communicated to a line 292 in meter 5 indicating the need to service an I/O request.

From a functional point of view, the signals on lines 280, 282, 285, 287, 290, and 292 are treated like switches, the status of which is periodically determined by microcomputer 90 during execution of the foreground routine in order to properly allocate task flow.

Fig. 7 illustrates the generation of certain signals within meter 5. A signal, designated $\overline{\text{POMODE}}$, is produced on a line 295 to reflect the status of mode changing switch 30. Mode changing switch 30 is magnetically coupled to a Hall Effect sensor 297 within meter 5. The normal state communicates a high level to line 297, indicating that the meter is in the "mail room mode". When switch 30 is placed in the "post office mode" position, sensor 297 communicates a low level to line 295.

The position of the print head is detected by a Hall Effect sensor 300 which cooperates with an iron element

301 on the printhead shaft. A signal, designated $\overline{\text{LEFT HOME}}$, is produced on a line 302, the level on line 302 being high when the print head is in its home position and low when the print head is away from its home position.

Fig. 8 includes a circuit schematic of power surveillance circuitry 105. Power surveillance circuitry 105 has the function of producing a signal, designated PWRLSS, on a line 330, whose level is high whenever VLOGIC is below about 9.5 volts. Line 330 is coupled to an output 335 of a voltage comparator 340. Comparator 340 is supplied at a first input 341 with VCC, and at a second input 342 with a precisely maintained fraction of VLOGIC. The fraction is defined by a voltage divider including a 5.9K, 1% resistor 345 and a 10K, 1% resistor 347. So long as VLOGIC remains above approximately 9.5 volts, the voltage at input 342 remains above 5 volts. VCC, being a regulated output voltage based on VLOGIC, remains at 5 volts so long as VLOGIC remains above approximately 7 volts. When VLOGIC falls below 9.5 volts, the voltage at input 342 falls below the voltage at input 341. Comparator 340 then causes a high level to appear at output 335 which is coupled to line 330. This PWRLSS signal is communicated to microcomputer.90 on line 330 to initiate a power loss interrupt.

The relative sequence of these events is best understood with reference to Fig. 10 which is a timing diagram illustrating a complete power up and power down cycle. As VLOGIC increases from 0 to 7 volts, VCC increases correspondingly from 0 to 5 volts. As VLOGIC increases from 7 volts to about 9.5 volts or above, VCC maintains its regulated 5 volt output. During the increase of VLOGIC from 0 to 7 volts, PWRLSS, which depends on VLOGIC which powers comparator 340, rises to assume a high level. Once VLOGIC has increased above 9.5 volts, PWRLSS assumes a low level, the transition being designated 355. Similarly, as VLOGIC falls below 9.5 volts, PWRLSS assumes a high level, the transition being designated 358. PWRLSS remains high as VLOGIC decreases from 9.5 to 7 volts, and thereafter generally follows VLOGIC as it decreases towards 0. VCC remains at 5 volts so long as VLOGIC is above 7 volts.

Transitions 355 and 358 in the PWRLSS signal are significant events that initiate a particular sequential generation of MPCLR and SYSCLR signals. This insures that operation of the electronic control system, including microcomputer 90, only occurs during periods when the available power is sufficient to ensure reliable operation of the components. Moreover, the sequence is designed to provide sufficient time to complete any updating of BAM registers before inhibiting microcomputer operations in order to maintain data integrity.

Broadly, SYSCLR, when high, inhibits operation of the print mechanism and writing data to BAMs 100 and 102. MPCLR, when high, inhibits operation of microcomputer 90. Accordingly, MPCLR must go low before SYSCLR goes low and remain low until after SYSCLR goes high in order to ensure that microcomputer 90 maintains control at all times that printing could occur or data could be written to the BAMs.

Referring to Fig. 10, the requirements on the timing of MPCLR and SYSCLR can be seen. Once PWRLSS has gone low, MPCLR must remain high for a duration 360 to ensure that microcomputer 90 has had power for a sufficient time to operate reliably. MPCLR then goes low, indicated schematically as transition 362. Microcomputer 90 then commences operation and SYSCLR, which has been high, then goes low, to allow normal meter functioning. This transition, indicated 365, must occur later than transition 362 by an interval 370. Similarly, when PWRLSS goes high, SYSCLR must remain low for a sufficiently long time to complete any writing to the BAMs that is in progress. A duration 372 must elapse before SYSCLR goes high, the transition being designated 375. After SYSCLR has gone high, MPCLR goes high, the transition being designated 377 occurring after transition 375 by an interval 380. The length of intervals 370 and 380 is of no particular significance, so long as the particular ordering is maintained. Intervals 360 and 372 are functions of the particular hardware configuration and components used. For the embodiment described herein, interval 360 is preferably greater than 50 milliseconds while interval 372 is greater

than 20 milliseconds. During interval 372 it is necessary that VCC remain at its 5 volt level to insure that the electronic components function reliably and that an enable on the BAMs (to be discussed below) remain above 2.2 volts. This means that VLOGIC cannot be allowed to fall below 8 volts in less than 20 milliseconds. This reserve capacity is provided by capacitor 201 (Fig. 5) which insures that enough power will be available to maintain reliable operation during interval 372.

Fig. 8 also includes a circuit schematic of system reset circuitry 110, which has the function of generating MPCLR and SYSCLR signals according to the above sequence in response to transitions 355 and 358 of the PWRLSS signal. System reset circuitry 110 actually provides complementary SYSCLR and MPCLR signals, designated $\overline{\text{SYSCLR}}$ and $\overline{\text{MPCLR}}$, on respective lines 390 and 392.

Circuitry 110 includes a comparator 400 having a first (positive) input 401 maintained at a fixed fraction of VCC by a voltage divider chain comprising 5.9K resistor 402 and a 10K resistor 404. A second (negative) input 406 is coupled to a capacitor 407 to ground and to a resistor 408 to VCC. The PWRLSS signal on line 330 communicates to comparator input 406 through a diode 409. An output 410 of comparator 400 communicates a circuit point 415.

Circuit point 415 is coupled through a resistor 417 to a circuit point 422 which communicates to different portions of the circuitry. First, circuit point 422 is coupled through a diode 425 to the base of a Darlington transistor 428, the emitter of which drives line 390 ($\overline{\text{SYSCLR}}$). The base of transistor 428 is coupled to a capacitor 430 to ground, and to VCC through a resistor 432, and further to the collector of a transistor 435. Second, circuit point 422 is coupled through a diode 440 to an intermediate circuit point 442 of a chain defined by a 75K resistor 445 to VCC, and a 4.7 microfarad capacitor 450 to ground. Capacitor 450 is coupled to the base of a Darlington transistor 455. The collector of transistor 455 is coupled through a diode 457 to a circuit point 458. Circuit point 458 is coupled through a

resistor 460 to the base of transistor 435, and through a resistor 465 to the base of a transistor 470, the collector of which is coupled to line 392 ($\overline{\text{MPCLR}}$).

Consider first a condition wherein a high level has persisted for a substantial length of time on line 330. Capacitor 407 is fully charged to VCC, and output 410 of comparator 400 is low. Accordingly, circuit point 415 is low, so that circuit point 422 is low. This causes the base of transistor 428 to be low, thereby causing transistor 428 to be in a nonconducting state so that the level on line 390 is low. That is, SYSCLR is high. The low level on emitter 422 maintains circuit point 442 at a low level, thereby keeping transistor 455 in a nonconducting state. Accordingly, circuit point 458 is high so that transistor 435 assumes a conducting state. The collector of transistor 435 is thus at a low level, thereby reinforcing the low level imposed through diode 425 across capacitor 430. Also, the high level at circuit point 458 turns transistor 470 on, and causes a low level to appear on line 392 (MPCLR is true).

When PWRLSS goes low (transition 355), the following events occur in order to cause MPCLR and SYSCLR to go low in the required sequence producing intervals 360 and 370. When the level on line 330 goes low, diode 409 conducts, thereby discharging capacitor 407 and causing the input 406 of comparator 400 to go low. Output 410 goes high, causing circuit point 415 to go high. A high level thus appears at circuit point 422. This blocks diode 440, thereby causing capacitor 450 to become charged through resistor 445. Resistor 445 is relatively large (75K) thereby resulting in a substantial delay before the voltage across capacitor 450 is enough to switch transistor 455 to its conducting state. This delay substantially defines interval 360, since once transistor 455 is conducting, circuit point 458 goes low, thereby turning off transistor 470 and causing a high level to appear on line 392. That is, MPCLR goes low. Once the delayed low level has appeared at circuit point 458, transistor 435 is turned off, thereby allowing capacitor 430 to become charged through resistor 432. This causes transistor

428 to become conducting after a time interval defined by capacitor 430 and resistor 432, at which point, the level on line 390 goes high. That is, SYSCLR goes low.

When PWRLSS has been low for some time so that the various components of power surveillance circuitry 110 are in the states described above, and then PWRLSS goes high (transition 358), the following sequence of events occurs. The high level on line 330 blocks diode 407, causing capacitor 408 to become charged through resistor 409. After a delay in excess of 20 milliseconds, the voltage on capacitor 408 exceeds the fractional level defined by resistors 402 and 404, and output 410 of comparator 400 goes low. This delay establishes interval 372. A low at output 410 causes a low at circuit points 415 and 422. This causes the base of transistor 428 to go low, thereby turning off transistor 428 causing a low to appear on line 390. That is, SYSCLR goes high.

When circuit point 422 goes low, diode 440 which was formerly blocked becomes conductive, causing capacitor 450 to discharge through resistor 447. After a delay determined by the time constant of capacitor 450 and resistor 417, transistor 455 is turned off thereby causing its collector to go high and conduction to occur through diode 457. This causes transistor 435 to conduct, thereby reinforcing the low at the base of transistor 428. In addition, the high level at circuit point 458 causes transistor 470 to conduct, thereby causing a low level to appear on line 392. That is, MPCLR goes high.

Fig. 9 is a circuit schematic of circuitry for generating signals to inhibit meter functioning upon detection of a fault condition. Two redundant circuits are employed, and with an exception to be discussed below, these two circuits are equivalent. Thus only one will be discussed herein.

One-shot 113 has an input 500 which is responsive to a falling edge. A resistor 506 and a capacitor 504 are coupled to one-shot 113 to establish basic timing. One-shot 113 has a reset input 507 to which the $\overline{\text{MPCLR}}$ signal on line

392 is coupled by a line 508. One-shot 113 has a complementary output 509 which is coupled to a first input of a NAND gate 510. NAND gate 510 has a second input to which is coupled the $\overline{\text{SYSCLR}}$ signal on line 390 by a line 511. NAND gate 510 has an output 512 which is coupled to flip-flop 111. Flip-flop 111 has an output 515 which is coupled through a diode 517 to the base of a Darlington transistor 520, the collector of which is coupled to line 390 of system reset circuitry 110 by line 511. Under normal conditions, microcomputer 90 periodically causes the generation of a signal, designated $\overline{\text{MPTS1}}$, to fire one-shots 113 and 114. As will be discussed below, the signal from microcomputer 90 causes a low level to appear at input 500. A low level is generated on complementary output 509 in response to a negative going level at input 500, and remains low for a duration defined by capacitor 504 and resistor 506. In the preferred embodiment this duration is 20 milliseconds. Since one-shot 113 is a retriggerable one-shot, a subsequent signal at input 500 extends the period that output 509 remains low. Thus, repeated signals at less than 20 millisecond intervals maintain output 509 low. So long as the level on output line 509 is low, the level on line 512 is high. Fault flip-flop 111 maintains a low level on output line 515 so long as the level on line 512 is high. If, for some reason, the level on line 512 goes low, a high level appears on flip-flop output 515, and remains, even if the level on line 512 goes high again. A high on line 515 blocks diode 517 and causes a high level to appear at the base of transistor 520. This causes a low level to appear on line 511, which low level causes the level on line 390 ($\overline{\text{SYSCLR}}$) to remain low regardless of the level of PWRLSS. Accordingly, once fault flip-flop 111 has been set to have a high level at its output 515, a low level is maintained on line 390. That is, once fault flip-flop 111 is set, SYSCLR remains high regardless of power up and power down cycling. By coupling $\overline{\text{MPCLR}}$ to one-shot reset input 507, the level on one-shot output 509 is caused to remain high after MPCLR is false until signals from microcomputer 90 appear at input 500. Also, by coupling $\overline{\text{SYSCLR}}$ to NAND

gate 510, a high level is maintained on line 512 when SYSCLR is true. Thus, fault flip-flop 111 is prevented from being set to its "faulted state" during the times that SYSCLR is true during normal power cycling.

Fault flip-flop 112 and retriggerable one-shot 114 are coupled in a similar way with the exception that one-shot 114 uses the non-complementary output on a line 540. The output on line 540 is coupled to both inputs of a NAND gate 542 which is physically removed from the integrated circuit chip including one-shots 113 and 114. Accordingly, if the chip on which one-shots 113 and 114 are located failed in a way that would cause all outputs to assume a high or low level, one of the fault flip-flops would be set. Only a dual failure which would cause the output on line 509 to remain low and the output on line 540 to remain high would go undetected. This is in keeping with the general meter design which can tolerate one failure and maintain integrity, but cannot accommodate certain combinations of compensating failures.

The level on line 390 is communicated to a circuit point 549 through cascaded Darlington buffers 545 and 547. Circuit point 549 is coupled to the cathode of fault indicator light 20, preferably a light emitting diode, the anode of which is coupled to VCC. Thus when circuit point 549 goes low light 20 is illuminated. Thus, light 20 is briefly illuminated during power up and power down cycling for those periods that circuit point 549 is low, providing the operator with an indication that the light works. Once SYSCLR has gone high, light 20 is illuminated. This occurs when a fault condition has occurred, regardless of power cycling.

Figs. 11a, 11b, and 11c, taken together, are a circuit schematic of microcomputer 90, BAMs 100 and 102, and circuitry for generating and responding to control signals related to accessing the BAMs.

Of the various control signals discussed above, microcomputer 90 reacts directly to PWRLSS and $\overline{\text{MPCLR}}$. A high level on line 330, signifying a low power condition is inverted by a NOR gate 560 and communicated to an interrupt

input 562 of microcomputer 90. A low level appearing at interrupt input 560 causes a power loss interrupt as discussed in connection with Fig. 8. The signal $\overline{MPCLR}$ on line 392 is communicated to a reset input on microcomputer 90. A low level on line 392 (MPCLR true) causes microcomputer 90 to be disabled and to cease execution.

Microcomputer 90 also generates control signals for selecting which of BAMs 100 and 102 is to be accessed, and which of I/O expanders 92 and 93 is to be accessed. These two selection procedures make use of the same data lines 132 and 133 (lines P24 and P25 of bus 130). The significance of the signals on lines 132 and 133 depends on whether microcomputer 90 is executing a background routine (BAM select) or the foreground routine (I/O expander select).

Microcomputer 90 selects one of I/O expanders 92 and 93 by causing the generation of signals designated PROG1 and PROG2 on respective lines 565 and 567. The level on output lines 132 and 133 is communicated to respective OR gates 570 and 571, the other input of each of which is connected to an output line 572 of microcomputer 90. The level on line 572 is normally high, but goes low at the same time that a control code for an I/O expander appears on line 134. The level on line 572 then goes high and at the same time data to be transferred appears on line 134. Thus the signal on line 572 appears on line 565 or 567, depending on whether line 132 or 133 is low.

Reading and writing to BAMs 100 and 102 are controlled by providing appropriate levels at four control inputs of each BAM. Each BAM includes two chip enable inputs 573 and 574, designated $\overline{CE1}$ and CE2, respectively. The level at chip enable input 573 must be low to enable access to the BAM contents, while the level at chip enable input 574 must be high. A low level at chip enable input 574 causes the BAM to assume a so called "low power mode" in which it draws only the small amount of power required to maintain its contents. In the low power mode, access to the BAMs in inhibited. Each BAM includes an output disable input 575, designated OD, which must be at a low level in order to read

the BAM. Each BAM includes a write enable input 576, designated $R/\overline{W}$, the level at which must be low for writing to occur, and high to disable writing during a read operation.

Microcomputer 90 selects which of BAMs 100 and 102 is to be read by causing a low level to appear on either of lines 132 and 133, which low level is communicated to respective chip enable inputs 573. Microcomputer 90 controls reading and writing to the BAMs according to the output level on a line 579 (line P27 of data bus 130). Line 579 communicates to a first input of a NAND gate 580, and the signal $\overline{\text{SYSCLR}}$ on the line 390 is communicated to a second input of NAND gate 580, so that the output of NAND gate 580 is low only when $\overline{\text{SYSCLR}}$ is high and the output on line 579 is high. The signal at the output of NAND gate 580 is communicated on a control line 585 to control inputs 576 on BAMs 100 and 102 to control writing to both BAMs.

The levels VLOGIC, VEXT1, and VEXT2 control reading and writing to the BAMs independent to microcomputer 90. VLOGIC communicates to respective chip enable inputs 574 on BAMs 100 and 102 through a common back biased 5.1 volt zener diode 590 and respective diodes 592 and 595. VEXT1 and VEXT2 also communicate to the respective inputs 574 through respective diodes 597 and 600. Thus when VLOGIC rises above about 5.7 volts, zener diode 590 breaks down and diodes 592 and 595 conduct. Thus the level at inputs 574 begins to rise and when it reaches about 1.5 volts, puts BAMs 100 and 102 into their so-called "high power mode" in which reading and writing may occur so long as other control signals are present. In the absence of VLOGIC, powering VEXT1 or VEXT2 allows the BAMs to be accessed. In the absence of sufficiently high levels of VLOGIC, VEXT1, and VEXT2, chip enable inputs are held low by pull-down resistors to ground.

Writing to the BAMs can occur independent of microcomputer control 90 for the purpose of allowing a postal service employee to zero certain locations when the meter has faulted. To permit this to occur, the signal $\overline{\text{POMODE}}$ on line 295 and a signal $\overline{\text{TWRITE}}$ on a line 609 are communicated to first and second inputs of a NOR gate 610. The

output of NOR gate 610 is communicated to the base of a transistor 615, the collector of which is coupled to BAM control line 585. Therefore, a low level on lines 295 and 609 causes a high level at the output of NOR gate 610, thereby turning transistor 615 on and causing a low level to appear on line 585, thereby allowing access to the BAMs.

In addition to controlling reading and writing to the BAMs, the $\overline{\text{SYSCLR}}$ signal on line 390 cooperates with signals from microcomputer 90 in order to generate a signal designated SON on a line 620 for controlling print head solenoid 85. Signals on an output line 622 (line P26 of bus 130) are communicated to a first input of a NOR gate 625. The signal on line 390 ($\overline{\text{SYSCLR}}$) is first inverted by a NOR gate 628, to produce a SYSCLR signal and communicated to a second input of NOR gate 625, the output of which is coupled to line 620. Therefore, a high level on line 390 and a low level on output line 622 are required for SON to be high. Otherwise, SON is low. As will be discussed below, SON must be high to allow the print head to rotate. The SYSCLR signal from NOR gate 628 is communicated to motor interface circuitry on a line 629.

Fig 12a is a circuit schematic of circuitry for interfacing microcomputer 90 to the switches, displays, BAMs, and motor position sensors. Data flow is established through I/O expander 92. I/O expander 92 receives signals on four bit data line 134 (lines P20 -P23 of data bus 130), in order to determine which of four data ports 650, 652, 654, and 656 is to be accessed, whether a read or a write operation is to be carried out, and whether the data is to be logically manipulated. Once this selection has occurred, data is communicated to or from microcomputer 90 on the same line 134. In the preferred embodiment, I/O expander 92 is used for generating address codes at its I/O ports in order to control demultiplexers 94 and 95 and multiplexers 96 and 97. Data port 650 is coupled to an address input 660 of multiplexer 96. Multiplexer 96 has 16 data inputs, not all of which are used. The following signals, the generation and significance of which were discussed in connection with Figs.

6 and 7 are communicated to respective inputs of multiplexer 96: $\overline{\text{POMODE}}$ on line 295; $\overline{\text{LEFT}}$ $\overline{\text{HOME}}$ on line 302; $\overline{\text{MB}}$ on line 280; $\overline{\text{TAPE}}$ on line 282; $\overline{\text{ENVELOPE}}$ on line 285; $\overline{\text{REPEAT}}$ on line 287; $\overline{\text{CLUTCH}}$ on line 290, and $\overline{\text{EXTINT}}$ on line 292. Selector switch 27 is coupled to two inputs on multiplexer 96 in order to communicate a low level to that input corresponding to the position of switch 27.

In addition to the above signals, signals designated $\overline{\text{KIN}}$ and $\overline{\text{RIN}}$ on respective lines 670 and 675 are communicated to respective data inputs of multiplexer 96. $\overline{\text{KIN}}$ and $\overline{\text{RIN}}$ are time multiplexed signals reflecting the state of selector switch 25 and keyboard 12. These will be discussed below.

Data port 652 is coupled via four-bit line 135 to an address input of multiplexer 97 and on a parallel line to BAMs 100 and 102. Data ports 654 and 656 communicate with demultiplexers 94 and 95 on respective data lines 680 and 685 for refreshing displays 15 and 17, and reading selector 25 and keyboard 12, as will be discussed presently. These operations cause the generation of $\overline{\text{KIN}}$ and $\overline{\text{RIN}}$ on lines 670 and 675.

Fig. 12a is a circuit schematic illustrating circuitry for periodically energizing the digits of displays 15 and 17, and reading keyboard switches 38, 40, and 42, and selector switch 25. Each digit of nine-digit display 17 comprises eight light emitting diode segments including seven bars for representing numeric symbols, and a decimal point. Display 17 includes a digit select input 695 for each digit (nine in all), and eight segment select inputs 698. Display 17 is a common cathode device, which means that a given digit select input 695 is coupled to the cathodes of all eight segments for that digit, while a given segment select input is coupled to the anodes of a particular segment for all nine digits. Thus, a low level at a particular digit select input 695 causes that digit to be illuminated according to which of segment select inputs 698 is at a high level.

Demultiplexer 94 includes ten data outputs 700, and a four-bit select input 702, such that a low level appears at

the output 700 corresponding to the binary code defined by the levels at select input 702. Nine of outputs 700 are connected in a one-to-one fashion to digit select inputs 695 on nine-digit display 17. Each of the ten demultiplexer outputs 700 is coupled to one pole of a numeric keyboard switch 38, the other pole of which is coupled to line 675 $\overline{RIN}$. Buffered data on line 129 is communicated through current limiting resistors 705 to a corresponding eight-bit line 707. Each individual conductor is coupled to one of segment select inputs 698 with one of the conductors, designated 708, coupled to the segment selector controlling the decimal points. Thus, according to the binary code on four bit line 680, one of multiplexer outputs 700 is set to a low level, which causes a low level to appear on line 675 if the corresponding keyboard key 38 was closed. Additionally, in the case of those nine of outputs 700 which are coupled to display digit select inputs 695, a low level causes that display digit to be illuminated according to the levels on line 129. Thus, each time one of the digits is energized, a corresponding signal indicates the status of a particular keyboard key. After the nine digits have been energized, the tenth demultiplexer output is selected, providing a level on line 695 that indicates the status of the tenth numeric keyboard key 38.

Four-digit display 15 comprises first and second two-digit displays 720 and 722, each digit of each display having eight light emitting diode segments including seven bars and a decimal point. Each digit includes a digit select input 730 and eight segment select inputs. For the particular display components in the preferred embodiment, each digit is independent, thereby requiring individual segment selection. Data on eight-bit line 707 is communicated to the segment selector inputs. Digit displays 720 and 722 are common anode devices, so that when the level at one of digit selector inputs 730 is high, that particular digit is illuminated according to which of the individual lines of line 707 is at a low level. Conductor 708 of eight-bit line 707 is coupled to the decimal point segment selectors for display

720, but two-digit display 722, corresponding to the least significant digits, has no connection to line 708. Rather line 708 is coupled to the cathode of indicator light 22, preferably a light emitting diode, the anode of which is coupled to that digit selector input 730 corresponding to the least significant digit of display 15.

Demultiplexer 95 has a four-bit select input 750, and ten data outputs, nine of which, designated 751 - 759, are connected as follows. Data outputs 751 and 752 are coupled to respective first poles of keyboard switches 40, and 42, respective second poles of which are coupled to line 675. Output 758 is coupled to a line 765 to provide a signal, designated $\overline{\text{EIO}}$ for controlling I/O peripheral devices 131. Output 759 is coupled to a line 767 to generate the signal $\overline{\text{MPTS1}}$ which is communicated to input 500 of fault one-shots 113 and 114.

Each of data outputs 753 - 757 is coupled to a contact of five position display selector switch 25, the common wiper of which is coupled to line 670. Of the five data outputs 753 - 757, four of them, 753 - 756, are coupled in a one-to-one fashion to digit select inputs 730. However, the coupling is not direct as in the case of the circuitry for energizing display 17, since display 15 is a common anode device. Each digit selector input 730 is coupled to the collector of a PNP transistor 765, the emitter of which is held at VDISP. Each of data outputs 753 - 756 is resistively coupled to the base of a corresponding one of transistors 765.

Accordingly, depending on the binary code defined by the levels on four bit line 685, a particular one of data outputs 751-759 is provided with a low level. When output 751 or output 752 goes low, a low level is applied to line 675 if special keyboard key 40 or 42, respectively, is depressed. If either of outputs 758 or 759 is selected, a signal having a falling edge followed by a low level is generated on line 765 or 767, respectively, communicating to other portions of the circuitry as described above. A low level on any of outputs 753 - 756 causes the corresponding

transistor 765 to become conducting, thereby providing a high level at the particular digit select input 730. Depending on which of the individual lines of line 707 is at a low level, the particular segments of the selected display digit will light. Noting that neither of the least significant digits ever requires a decimal point following, indicator light 22 is connected in a manner that substitutes for the decimal point of the least significant digit whose decimal point cathode is not connected to anything. Thus, when the least significant digit is selected and the level on line 708 is low, "Add \$" indicator light 22 is energized. Also, data outputs 753 - 756, in cooperation with data output 757, provide the signal $\overline{\text{KIN}}$ for communicating the status of switch 25 to microcomputer 90.

Fig. 13 is a circuit schematic illustrating the circuitry for interfacing microcomputer 90 to stepping motors 45 - 48 and position sensors 75 - 78. While Fig. 13 illustrates the complete interface circuitry, only one stepping motor 47 and corresponding position sensor 77 is shown for clarity. Broadly, the interface circuitry includes I/O expander 93, multiplexer 97, and a plurality of Darlington buffers 780 coupled to VMOTOR to provide buffered outputs that are either low or at a level VMOTOR. I/O expander 93 includes four data ports 785, 786, 787, and 788, signals at which are communicated through Darlington buffers 780 to respective motors 45, 46, 47 and 48, with port 787 communicating with motor 47. I/O expander 93 has an enable input 790 to which SYSCLR is communicated via line 629, and a program input 792 to which PROG1 is communicated on line 133. Control signals and data are communicated in a time multiplexed fashion on four-bit data line 134.

Motor 47 may assume one of ten positions over its angular range that correspond to proper print wheel settings, and eleven positions halfway between. Thus the possible positions can be broken into four groups including a group of so-called "odd" positions, a group of so-called "even" positions, and two groups of so-called "half" positions, each position of which is halfway between an even and an odd position.

Motor 47 includes four windings, corresponding to the four groups of positions. In general, the motor can only be stepped to an adjacent position by energizing the winding corresponding to the group of which the adjacent position is a member. One side of each winding is coupled to VMOTOR in the meter base through a current limiting resistor 789 located in the base. The other side of each winding is coupled to one of the buffered outputs corresponding to the data lines from port 787. A low level on a line at data port 787 turns off the buffered output, thereby resulting in no voltage across the particular winding. Only when a high level appears on an output line at port 787 is a low level communicated to the motor winding, thereby energizing it.

Since energization of a motor winding generally only causes the motor to step to an adjacent position, and since only four types of positions are recognized, absolute position determination may only be established by first driving motor 47 to one of its end points, and thereafter keeping track of all step commands. This is done during execution of the initialization routine.

Position sensor 77 comprises four sets of contacts 791, 792, 793, and 794, each set including ten sequentially spaced contacts, and four grounded wipers 801, 802, 803, and 804, each associated with a respective contact set. Wipers 801 - 804 are mechanically coupled to the mechanical output of motor 47, so that at a given integral position of motor 47, each of wipers 801 - 804 makes electrical connection with one of the contacts in its respective contact set 791 - 794. A binary code representative of the position of wipers 801 - 804 is generated on four corresponding output lines 811, 812, 813 and 814, one end of each output line being coupled only to particular individual contacts within its respective contact set. For example, the "1" bit is generated on line 811 by coupling line 811 to every other contact in set 791; the "2" bit by coupling line 812 to every other group of two contacts in contact set 792. The other end of each contact line is coupled to a data input of multiplexer 97. Four-bit data line 135 from port 652 of I/O expander 92 is coupled to

an address input 820 of multiplexer 97. According to the binary code represented by the levels on line 135, one of the four bits of information for one of the four position sensors appears at the output of multiplexer 97, and is communicated on line 99 to microcomputer 90.

Print head solenoid 85 prevents movement of the print head away from its home position unless the solenoid is energized. Control is established by the signal SON on line 620. Solenoid 85 includes input terminals 824 and 825. Input terminal 824 is coupled to VMOTOR in the base through current limiting resistors 827 and 828 located in the base. A capacitor 829 is connected across resistor 828 to provide greater initial current. The signal SON is communicated to the base of a Darlington transistor 832, the emitter of which is grounded and the collector of which is coupled to solenoid input terminal 825. Thus, a high level on line 620 causes a low level to appear at input terminal 825 of solenoid 85, thereby energizing the solenoid to permit movement of the print head. A diode 835 is connected across solenoid 85 to protect Darlington transistor 832 from overvoltage when SON goes low.

In view of the above description covering the general operation and detailed construction of meter 5, a description of the detailed operation of microcomputer 90 may be understood. The following discussion and the accompanying figures provide additional detail with respect to the discussion with reference to Fig. 4 outlining the general operation of microcomputer 90.

The operating program for microcomputer 90 is stored in 2,048 bytes of ROM while working memory is provided by 64 bytes of random access memory (hereinafter RAM). Fig. 14 illustrates in tabular form a suitable RAM allocation for storing information required by the operating program of microcomputer 90. For ease and consistency of nomenclature, the individual memory locations will be referenced in first instance with respect to their hexadecimal address within the 64 bytes of RAM, and subsequently by reference to appropriate mnemonics.

Location 0-7 Hex, designated R0-R7 and locations 18-1F Hex, designated R0'-R7' are two sets of directly addressable working registers.

Registers R0-R7 are allocated to the background tasks, to be described more fully below. Register R2 is dedicated to the background status and is hereinafter designated BAKSTA. The individual bits are shown schematically in Fig. 16, and include the power loss bit and the various WAKEUP bits set by the foreground to signify to the background dispatcher loop that a particular task is to be carried out. The particular significance of the various WAKEUP bits will be discussed with reference to the foreground routine and the appropriate background tasks described below. The individual bits in BAKSTA are sequentially checked by the background dispatcher loop as described above with reference to Fig. 4.

Locations 34 Hex and 35 Hex, designated KEYSTA and PRSTA respectively, are used to designate status for the keyboard and printer tasks. Locations 38-3B Hex, designated STEPIN(0)-STEPIN(3) and locations 3C-3F Hex, designated STEPTK(0)-STEPTK(3) provide an additional data base for the print routine. Location 36 Hex, designated PRCTR, is used to store a counter which, when set to a non-zero value, is decremented at each entry into the foreground routine in order to keep track of the time that has elapsed since various print subtasks were initiated. Locations 8-13 Hex provide 12 bytes for a six level stack to allow for six levels of sub-routine calls.

As discussed above, the foreground routine is executed in response to periodic timer interrupts, that occur at 400 microsecond intervals. The main functions of the foreground routine are energizing the digits of four-digit display 15 and nine-digit display 17, ascertaining the status of various switches and signals (collectively referred to as switches), debouncing portions of the switch information, and triggering one-shots 113 and 114. These functions are carried out by selecting I/O expander 92 and setting up appropriate binary codes on four-bit line 134 to communicate with

the appropriate switch or display digit via demultiplexer 94, demultiplexer 95, or multiplexer 96.

Registers R0'-R7' are the working registers used by the foreground that enable the foreground routine to carry out its functions according to the desired sequence. For example, four-digit display 15 is sequentially energized four times for every time that nine-digit display 17 is energized in order to provide adequate digit intensity. Register R3' is a counter for keeping track of this sequencing. Register R2' is a pointer that indicates which display digit is currently being done, while R1' is a basic pointer.

Locations 20-28 Hex and 30-33 Hex are used to store the display segment codes for the current values of the digits to be displayed on nine-digit display 15 and four-digit display 17, respectively. The foreground routine sequentially places these segment codes on bus 125 (P1) as the digits are sequentially selected through demultiplexers 94 and 95.

Switch information is stored in four switch registers. Location 1F Hex, designated R7FORG, is dedicated to certain of the base syncronization signals and other signals received through multiplexer 96, the status of which is desired without debouncing. Location 2A Hex, designated REG07, location 2C Hex, designated REG89, and location 2E Hex, designated REGSW, reflect the current debounced state of all the switches that can be set by the operator. The respective bit allocations for switch registers R7FORG, REG07, REG89, and REGSW are shown in Fig. 15. Locations 2B Hex, 2D Hex, and 2F Hex are so-called LAST TIME CHANGE registers corresponding to switch registers REG07, REG89, and REGSW, respectively and carry information indicating which switches corresponding to the bits of the corresponding switch register had changed status on the previous reading by the foreground.

As discussed above, switches are debounced in order to eliminate the effects of switch noise, and the status of a switch to be debounced is considered to have changed only if such change has persisted for two successive readings.

Debouncing of the switches is done in two steps. Preliminary debouncing is done on a switch-by-switch basis as each switch is read while final debouncing occurs when all the switches in a given switch register have been read and preliminarily debounced. As discussed above, nine of the ten keyboard switches 38 and four of the five positions of register display selector switch 25 are each read at the same time that a particular display digit is energized. Thus, each one of these switches is read on a separate pass through the foreground routine in conjunction with the energization of a particular display digit. Other switches are read serially in groups during particular single passes through the foreground routine.

Preliminary debouncing occurs as each switch is read. During preliminary debouncing, the switch register for the group of switches being debounced is copied to working register R4', and the corresponding LAST TIME CHANGE register is copied to working register R5'. R0' points to the current one of the switch registers that is being debounced. Preliminary debouncing on the group of eight switches in the particular register stored in R4' occurs over a number of foreground entries. As each switch is read on line 98 from multiplexer 96, the switch state is compared with the corresponding bit in the debounced switch register currently copied into register R4'. A corresponding bit is set in register R6' if the switch state read represents a change from the debounced value in register R4'. When all eight switches in the register being debounced have undergone this preliminary debouncing, R6' contains a record of those switches whose state at the most recent reading has changed with respect to the debounced values. This is stored as the LAST TIME CHANGE register for the next pass of the foreground. Final debouncing occurs by comparing registers R5' and R6'. If, for a particular switch, a change has persisted for two successive readings (corresponding bits in registers R5' and R6' being set) the corresponding bit in the debounced switch register is set to reflect the newly debounced state.

Fig. 17 is a flow chart of the foreground routine illustrating logical branches for executing a preferred sequence for servicing displays 15 and 17, ascertaining the status of the various switches, and performing debouncing. Upon the occurrence of a timer interrupt, microcomputer 90 enters the foreground routine and performs entry tasks generally designated as block 850. These include saving the background accumulator, restarting the 400 microsecond timer, and checking the value of PRCTR (the printer counter). If PRCTR is non-zero, PRCTR is decremented by 1 and the TIMER WAKEUP bit is set if PRCTR has reached zero. The state of bus 130 is saved in register R1', and lines 132 and 133 are set up to select I/O expander 92 for the purpose of reading switches.

The routine then checks whether the foreground is enabled (logical branch 852). If the foreground is disabled, the routine triggers the one-shots by setting the appropriate binary code on demultiplexer 95 in order to select output line 767 (block 855). The routine then restores the background bus 130 and accumulator and returns (block 857).

If the foreground is enabled, the particular display digit is energized according to a preferred sequence. According to the preferred sequence, the digits of nine-digit display 17 are sequentially energized on successive entries to the foreground routine, and then the digits of four-digit display 15 are sequentially energized. The energization of display 15 occurs four times for every time display 17 is energized. In order to maintain this sequence, the foreground routine maintains bookkeeping information in foreground working registers R0'-R3'. This bookkeeping information includes pointers for addressing the particular switch register being debounced, and counters for determining which digit in a given display is to be energized and which of the four passes through the four digits display is being carried out.

If at logical branch 852 the foreground is found to be enabled, the routine checks which display is being serviced (logical branch 860). If the nine-digit display is

currently being serviced, the routine checks if either the first or ninth keyboard key is to be read (branch 862). If so, the routine sets up registers R4' and R5' with REG07 or REG89, respectively, and the associated LAST TIME CHANGE register (block 865). Selecting one of the first nine keyboard keys ("0" through "8") to be read occurs in connection with the energization of one of the nine digits of display 17 (block 867). Reading of the tenth keyboard key ("9") is done independently of a digit energization.

During the first eight passes, the numeric keys corresponding to "0" through "7" are being sequentially read, and working register R6' accumulates information relating to changes in these switches for REG07. On the eighth pass (corresponding to keyboard key "7"), final debouncing of REG07 occurs (block 872). A change in REG07 is flagged by setting bit 7 of register R7FORG. The foreground routine then prepares for debouncing switch register REG89 on subsequent foreground passes (block 873), and returns to the background (block 857).

If at logical branch 870 it is ascertained that the eighth keyboard key ("7") is not to be read, a separate test is made to determine whether the tenth key ("9") is to be read (logical branch 875). If the tenth key is not being selected, as would be the case most of the time, i.e. either during preliminary debouncing of REG07 or of the ninth key for REG89, the pointer in working register R2' is incremented (block 877), and microcomputer 90 returns to the background.

If at logical branch 875 it is determined that the tenth key is to be read, that key is read, and the additional switches for REG89 are sequentially read on the same foreground pass and preliminarily debounced (block 880). REG89 is then finally debounced (block 882). The foreground routine then sets bit 7 of REG89 if the meter is not a fractional cents meter or if switch 30 is set to the "post office" mode position or if switch 27 is in the .00 position, in order to provide information for the keyboard task (block 885). If a change has occurred to REG89 or REG07 (as remembered by R7FORG(7)), the KEYBOARD WAKEUP is set (block 887).

The foreground routine then reads those signals that are to be stored in register R7FORG (block 890). If EXTINT is set, the I/O WAKEUP is set (block 892). In the event that the newly read LEFT HOME or CLUTCH signal is different from the corresponding formerly read signal, the HOME/CLUTCH WAKEUP is set (block 895). The pointers are then set so that the next entry into the foreground routine will cause servicing of four-digit display 15 (block 897).

When it is ascertained at logical branch 860 that four-digit display 15 is to be energized, a test is made as to whether the first position of display selector switch 25 is to be read (logical branch 900). If so, the switch registers are set up to debounce register REGSW (block 865). Selecting one of the first four of the five positions of switch 25 to be read occurs in connection with the energization of one of the four digits of display 15 (block 902). Reading the fifth switch position is not accompanied by selection of any of the four digits on the display. If the fifth position is not being read, the routine branches at logical branch 905 to block 877 and then returns to the background. On the fifth position, the routine tests whether the four-digit display has been energized four times (logical branch 907). If not, register R2' is reset so that the next pass will read the first position (i.e. select first digit) (block 908). The routine then branches to test whether the foreground is enabled (branch 852). The foreground will be found to be enabled (having previously been found so on the same foreground pass), so the first position is read on this pass as the routine branches as described above. If it is ascertained that the display has been energized the required four times, the fifth position is preliminarily debounced and the remaining switches whose bits make up REGSW are read and preliminarily debounced (block 910), and register REGSW is finally debounced (block 912). If REGSW has changed, the TAPE/ENVELOPE WAKEUP is set if the change is to either of the bits corresponding to the TAPE or ENVELOPE signals (block 915). The REGISTER DISPLAY WAKEUP is set if the change is to one of the register select switches (block 920). After

performing bookkeeping to prepare for servicing nine-digit display 17 on the next foreground entry (block 922), the routine then fires the one-shots (block 855) and returns to the background.

Fig. 18 is a memory map showing a preferred organization of BAMs 100 and 102, hereinafter sometimes referred to as BAM1 and BAM2, respectively. BAM1 and BAM2 are organized wherein a given hexadecimal address refers to a given hexadecimal digit (4 bits, hereinafter referred to as a nibble).

The key feature in a postage meter, electronic or mechanical, is the maintenance of accounting information to insure that all postage printed is paid for. As discussed above, microcomputer 90 maintains a descending register indicating the amount of postage still remaining (generally designated DR), an ascending register indicating the amount of postage printed (generally designated AR), and a control total which should remain constant between additions of postage at the post office (generally designated TOTAL). In keeping with the fault tolerance aspects of the present invention, these registers, redundant among themselves, are stored in redundant BAMs 100 and 102, the dual copies being designated DR1, AR1, and TOTAL1 for BAM1 and DR2, AR2, and TOTAL2 for BAM2, respectively.

Additional accounting information for use by the user is maintained in dual redundant user resettable registers designated BATCH COUNT and BATCH TOTAL. Each time postage is printed, these registers are respectively incremented by 1 and by the amount of postage printed. Resetting of these registers is effected by pressing "clear batch" key 42 simultaneously with "clear keyboard" key 40. A register designated COUNT is incremented by 1 on each print cycle to provide maintenance information for repair personnel at the factory by indicating the total number of print cycles the meter has undergone. The COUNT register is not accessible to the user.

DR1, AR1, and TOTAL1 are stored at nibbles 00-09 Hex, 10-19 Hex, and 20-29 Hex, respectively in BAM1, with DR2, AR2, and TOTAL2 being stored in corresponding locations

in BAM2. BATCH TOTAL 1 & 2, BATCH COUNT 1 & 2, and COUNT 1 & 2 are stored at nibbles 30-39 Hex, 40-49 Hex, and 50-59 Hex, respectively in BAM1 and BAM2. Register display selector switch 25 allows the user to display on nine-digit display 17 one of the registers DR, AR, TOTAL, BATCH TOTAL, and BATCH COUNT. A keyboard register, designated KB, is stored at nibbles 60-69 Hex.

Temporary copies of DR1, AR1, designated TDR1, and TAR1, respectively are stored in nibbles 70-79 Hex and 80-89 Hex, of BAM1. Corresponding copies designated TDR2 and TAR2 are stored in corresponding locations in BAM2. However, as will be described below, only TDR1 and TAR1 are updated during keyboard entry. Nibbles 90-99 Hex in each of the BAMs are used as a temporary register, designated TEMP1 and TEMP2.

Each of the registers described above contains ten nibbles, nine nibbles of which contain the numerical value in BCD representation. The tenth nibble of each register is called the dirty register nibble, bit 3 (most significant bit) of which is set to "1" before a register to register move is carried out into that register. The dirty register nibble is then zeroed at the end of the move.

All the registers except COUNT contain the information to three decimal places. Thus incrementing BATCH COUNT by 1 is carried out by adding 1000 which is thought of as 1.000. COUNT is stored as a whole number and therefore is incremented by adding 1. Nibbles A0-AC Hex of BAM1 are used to store a BCD 1000 which is designated REGTH01 when addressed as nibbles A0-A9 Hex and as REGONE1 when addressed as nibbles A3-AC Hex. Corresponding locations in BAM2 are designated REGONE2 and REGTHO2.

If microcomputer 90 discovers a condition requiring that the meter be set to a faulted condition a fault code is first written into nibble 2D Hex of both BAMs so that subsequent examination of the BAM contents can tell the factory what happened. The particular hexadecimal codes for various fault conditions are as follows:

F Hex - An error is detected after writing a nibble to a BAM and reading and verifying the newly written nibble.

7 Hex - An arithmetic error is detected when adding a new character to the keyboard register according to the checked arithmetic algorithms to be described below.

6 Hex - A disagreement between the corresponding permanent registers in BAM1 and BAM2 is detected either before retrieving the register value or after updating the register value.

3 Hex - The print head leaves its home position before positioning of the print wheels is completed, indicating a problem with solenoid 85 or home sensor 300.

2 Hex - The print head does not leave its home position within 100 milliseconds of receiving the debounced clutch signal, indicating a problem with home sensing switch 300.

Nibble 2E Hex of both BAMs is initialized at the factory and is used to designate the maximum number of digits that can be entered into the keyboard register while meter 5 is in the "post office" mode. Nibble 2F Hex of both BAMs is initialized at the factory and describes the type of meter as follows. A "1" in Bit 1 indicates that the meter is a United Parcel Service (UPS) meter which has no lockout when the descending register goes negative. A "1" in Bit 2 indicates that the meter is a fractional cents meter. A "1" in Bit 3 indicates that there are four rather than three stepping motors. Nibble 5A of BAM1 is used to store a keyboard character counter in order to tell the keyboard register routine when the keyboard register is full. Bit 0 is always "1."

Background operations that have an effect on DR, AR, or TOTAL are carried out according to checked arithmetic algorithms. There are three different arithmetic operations, the addition of postage in the "post office" mode, the decrementing of DR and corresponding incrementing of AR during a normal printing cycle, and a checking routine used during initialization, keyboard entry, and clearing the keyboard.

The basic premise of all the checked arithmetic algorithms is that the descending register DR and the ascending register AR must sum to the control total. In keeping

with the fault tolerance aspect of the meter, there is one copy of each of these three registers in each BAM. During arithmetic operation, these registers are moved to the temporary registers in the BAMs described above. Prior to such a move, a verification is made that the contents of corresponding BAM registers are equal.

The contents of the temporary registers at the completion of various routines and tasks are shown in tabular form in Fig. 19. The particular sequences giving rise to these final states are now described.

A subroutine designated CHKTOT, carries out a number of the above verifications in three places - during initialization, during a keyboard clear, and while adding to the keyboard register in "mail room" mode. Subroutine CHKTOT checks both copies of the three permanent registers (DR, AR, TOTAL) to make sure that they are equal, moves them to temporary locations and checks that their temporary location values hold to the equation TAR + TDR = TTOTAL (TEMP2). It would require an extremely unlikely occurrence of two errors exactly complimenting each other to have a wrong TAR, TDR, or TTOTAL. As will be seen below, the other checked arithmetic algorithms carry out manipulations on TDR1, TDR2, and/or TEMP1 and thereafter update the permanent registers in both BAMs with these values. In order to facilitate reconstruction should microcomputer 90 cease to function properly before all updated registers have been copied into their permanent location, the non-updated descending and ascending registers are maintained in TDR2 and TAR2 respectively. The operation of CHKTOT is as follows:

1. DR1 and DR2 are compared.
2. DR1 is moved to TDR1 and TDR2.
3. TOTAL1 and TOTAL2 are compared.
4. TOTAL1 is moved to TEMP1 and TEMP2.
5. AR1 and AR2 are compared.
6. AR1 is moved to TAR1, TAR2, and TEMP1.
7. TDR1 is added to the copy of AR1 in TEMP1 and stored in TEMP1.

8. TEMP1 and TEMP2, which should both equal TOTAL are compared.

Although the print task and keyboard task routines will be described in detail below, the checked arithmetic aspect of those tasks will be described at this point. As each character is entered, a keyboard register, designated KB, is updated to reflect the entire entry since the last keyboard clear. The basic algorithm for keyboard entry is that each character entry causes new temporary values of AR and DR to be calculated, being given by AR + KB and DR - KB, respectively, and to be stored in TAR1 and TDR1, respectively. The updated temporary register values are checked to make sure that they sum to TOTAL. The particular sequences of steps for updating the temporary registers upon entry of a keyboard character is as follows.

1. The keyboard register KB is updated to reflect the total keyboard entry.

2. CHKTOT is called to get new copies of AR1 in TAR1 and TAR2 and of DR1 in TDR1 and TDR2.

3. KB is added to the copy of AR1 stored in TAR1, and the result is stored in TAR1.

4. KB is subtracted from the copy of DR1 stored in TDR1, and the result is stored in TDR1.

5. The updated TAR1 is moved to TEMP1.

6. The updated TDR1 is added to TEMP1 and the result stored back in TEMP1.

7. TEMP1 and TEMP2 (the latter having a checked copy of TOTAL) are compared to make sure that they are equal.

In a print cycle, updating of the permanent registers occurs after the print head has left home, or is expected to leave home as evidenced by either the LEFT HOME signal or the receipt of CLUTCH signal. Updating occurs as follows:

1. TDR1 is copied into DR1 and DR2.

2. TAR1 is copied to AR1 and AR2.

3. Batch and count arithmetic is done.

4. After returning to home, CHKTOT is called to check that the updated permanent registers are equal and hold to the required arithmetic constraint.

Additional checked arithmetic procedures are carried out when the meter is in the "post office" mode. When the meter is in the "post office" mode, the temporary registers are not updated during keyboard entry as described above in connection with the normal keyboard entry prior to a print cycle. Thus, prior to a change by the postal service employee, the state of the temporary registers is like that immediately following a call to CHKTOT. That is, the temporary registers have the current permanent values of the ascending and descending registers and the total.

As discussed above, special keys 40 and 42 allow the postal service employee to change the value of DR and TOTAL by adding or subtracting a corresponding amount (the keyboard register) from both. Thus, depression and release of clear batch key 42 causes the keyboard entry to be added while if clear KB key 40 is depressed while key 42 is also depressed, release of key 42 causes the keyboard entry to be subtracted. The basic arithmetic constraint is that the updated descending register (DR $\pm$ KB) when added to AR must equal TOTAL $\pm$ KB. Upon releasing clear batch key 42, the following steps take place.

1. KB is added/subtracted to/from TDR1 and the result stored in TDR1.

2. A check is made that DR will not be greater than $99,999.99 or less than zero.

3. KB is added/subtracted to/from TEMP1 (which contains TOTAL).

4. TEMP1 is moved to TEMP2.

5. TDR1 and TAR1 are added and the result stored in TEMP1.

6. TEMP1 and TEMP2 are compared.

7. TDR1 is moved to DR1 and DR2.

8. TEMP1 (new total) is copied to TOTAL1 and TOTAL2.

9. CHKTOT is called to insure that the moves were done without error.

In view of the above description of the foreground routine and the checked arithmetic algorithms, the initial-

ization routine can be understood. The initialization routine is executed when MPCLR goes low in a power up cycle to permit operation of microcomputer 90, and performs those functions necessary to bring meter 5 into operating condition in a valid, known state. These functions may be summarized as follows:

1. Solenoid 85 is turned off (SON low) to inhibit printing unless specifically carried out under control of microcomputer 90.

2. The memory locations in RAM, the memory allocation of which is illustrated in Fig. 14, are set to zero.

3. The foreground registers are initialized with the foreground disabled so that the foreground routine will start triggering one-shots 113 and 114.

4. Timer PRCTR is set to provide an interval of 100 milliseconds, to allow the power to stabilize. Then the timer and its interrupt are enabled. After the 100 milliseconds the power loss interrupt in microcomputer 90 is enabled. By this time power is stable and SYSCLR is false and operation of the meter commences.

5. A check is made that the fault nibbles in BAMs 100 and 102 are zero. As discussed above, when meter 5 is set to a faulted condition, it should be impossible to bring the meter up in a power up cycle. Accordingly, detection of a non-zero value here indicates either a lack of ability to read a good zero from the fault nibbles, or a malfunction of the system reset circuitry or of the BAMs themselves. If a non-zero value is found, the meter is set to a faulted condition but no specific error code is written.

6. The information relating to meter type and maximum digit entry is recovered from the BAMs and written to registers in RAM for ready availability to the operating program.

7. The keyboard is cleared by executing a clear keyboard routine which in turn calls CHKTOT to check that the BAM contents are equal and hold to the arithmetic constraint.

8. REGONE1 and REGONE2, and REGTHO1 and REGTHO2 are checked to make sure that they are respectively equal.

9. The foreground is enabled for 100 milliseconds to update the switch registers.

10. The foreground is disabled and a check is made to see if the meter is on a base ($\overline{MB}$ low). If the meter is not on a base, the keyboard is cleared and the background dispatcher loop is entered where it is expected that the register display WAKEUP bit will be set.

11. Assuming the meter is on a base, the printer registers are set up and the stepper motors are stepped to 9999 without verification by sending out more step commands than are necessary. This provides for the possibility that the stepper motors are stiff or sloppy after a period of disuse. Once it is determined that the print wheels are at 9999, the meter is stepped to 0000 with verifying. The meter continues to try until reaching 0000, and will loop to the beginning of step 11 above if it can't. The background dispatcher loop is then entered, with the register display and keyboard WAKEUP bits set so that the keyboard register and selected register will be displayed.

Figs 20a, 20b and 20c, taken together, form a flow chart of the print routine for controlling postage printing and monitoring the operation of the print mechanism. Security is maintained by requiring the various events in a print cycle to occur in a well-defined sequence, and within predetermined time intervals. The meter is set to a fault (or soft fault) state if any of the relevant signals are inconsistent with the meter's being in a known and expected state. Broadly, the print routine positions stepper motors 45-48 (and their respective print wheels therewith) on receipt of a TAPE or ENVELOPE signal. When positioning is completed, solenoid 85 is turned on, and the CLUTCH and/or LEFT HOME signal is awaited. When the LEFT HOME signal indicates that the print head has left the home position or the CLUTCH signal indicates that the clutch has been pulled, AR and DR are updated from temporary locations in the BAMS and the batch and count registers are updated. Upon a return of the print head to its home position, the solenoid is then turned off. If only a single tape is to be printed the

keyboard register is cleared. Either way, new temporary values (TAR = AR + KB, and TDR = DR - KB) are calculated.

During execution of the print routine keyboard entry and receipt of I/O signals are blocked. If while printing in response to an ENVELOPE signal, another ENVELOPE signal is received, the print routine does not end upon the print head's returning to the home position, but rather leaves the solenoid energized and calculates a new value of TAR and TDR. The print routine then leaves itself in a state where it is waiting for the CLUTCH and/or LEFT HOME signal after as if it had just completed positioning.

Before discussing the operation of the print routine in detail, reference should be had to Fig. 16 which illustrates schematically a portion of the data base from which the routine operates. Each stepper has two registers allocated to it, one designated STEPIN(X) and one designated STEPTK(X) where X is the number of the stepper. The most significant nibble of STEPIN carries the instruction indicating which of the four windings is to be energized. The least significant nibble of the STEPIN contains a binary decimal code for the Final Position the particular stepper motor is to assume prior to printing. This is normally determined from the appropriate digit of the keyboard register. The least significant nibble of STEPTK contains the Next Position to which the stepper is to be stepped, while the most significant nibble of STEPTK carries separate bits of information, indicating the direction to step, whether the present position (verified by reading the switches) is equal to Next Position, and also if equal to Last Position, whether a second try is being made to reach the next position, and whether the next position expected is a half position.

A register designated PRTSTA carries separate bits or groups of bits indicating which stepper is currently being stepped, the current printer task as will be described in detail below, whether the initialization routine is being executed, and the status of certain bits from REGSW (TAPE, ENVELOPE and REPEAT). Additionally, reference is made to those bits in R7FORG signifying the status of the MB, LEFT HOME, and CLUTCH signals.

The print routine is responsible for performing multiple functions at different times during a print cycle. In order to maintain a proper sequence, the routine performs tasks in a fixed order. At the completion of a given task, the routine updates the relevant bits in PRSTA so that subsequent entry into the routine will cause the next task in sequence to be carried out. The order of the tasks, and the numerical code corresponding to the different tasks are as follows:

Task 0 - This is the state at the initiation of a print cycle and indicates that no task is pending. During task 0, the routine builds up portions of the required data base.

Task 1 - The routine sends out the most recently executed commands (from the previous print cycle) in order to make sure that the positions of the steppers correspond to those that are known (i.e. stored in STEPIN). This is done to take account of the possibility that one or more of the steppers has been jarred away from its last position since the last positioning operation.

Task 2 - The routine verifies that the steppers are in fact at their last position. Then the keyboard digits are copied to STEPIN to signify the final positions to be achieved, and the appropriate directions of stepping are chosen.

Task 3 - The routine sends out commands at 1.6 millisecond intervals to cause the steppers to step to their final (keyboard) positions.

Task 4 - This task is carried out once the stepper positioning has been completed. The routine is waiting for a signal indicating that the clutch has been pulled and/or that the print head has left its home position. If the CLUTCH signal is received first, the routine sets a 100 millisecond timer, within whose interval the LEFT HOME signal from the home sensor must be received. Once the print head has left its home position or the clutch is pulled, the BAM registers AR and DR are updated to reflect the amount of postage being printed on this cycle and the BATCH and COUNT registers are updated.

Task 5 - This task checks that the CLUTCH signal has gone away and that the print head has returned to its home position. At this point, the task is set to 0.

Tasks 6 and 7 - These tasks are carried out rather than tasks 4 and 5, respectively, if a new envelope signal is received once positioning has been carried out. Upon reaching task 7, the task is set to 4 to prepare for another printing operation.

Referring to Fig. 20a, the sequence of the print routine may be understood. It should be recognized that during a print cycle, the print routine may be entered multiple times in response to the TAPE/ENVELOPE WAKEUP, the HOME/CLUTCH WAKEUP or the TIMER WAKEUP. The routine immediately checks whether the meter is in the "post office" mode (logical branch 930), in which case the meter soft faults (block 932). Finding the meter to be in the "mail room" mode, the routine disables the foreground (block 935), checks that there are sufficient funds remaining (logical branch 937), and checks to determine whether all the steppers are at their final positions (logical branch 940). Assuming the positioning has been completed, the routine carries out tasks 4-7 as will be discussed below.

Prior to carrying out tasks 0-3 which effect positioning of the print wheels, the routine checks to make sure that the HOME/CLUTCH wakeup bit is not set (logical branch 942). If the HOME/CLUTCH WAKEUP is found to be set, the routine checks the cause (logical branch 945). An indication from the LEFT HOME signal that the print head has left its home position prior to the completion of positioning evidences a situation in which meter security is compromised since solenoid 85 or home sensor 300 may be malfunctioning. This is responded to by causing the meter to fault with hexadecimal code 3 (block 947). The presence of an unexpected clutch signal is generally caused by a base malfunction or improper paper handling and does not represent a threat to security. However operation must be suspended, and to this end the meter soft faults (block 950).

Assuming the HOME/CLUTCH WAKEUP is not present, a three way branch is made according to the current task (block 952). If the current task is 0, as is true on the first pass through the print routine during a particular print cycle, a "PRINTER ONLY" bit is set to inhibit the keyboard and I/O, relevant switch data from REGSW is loaded into PRSTA, and the task is set to 1 (block 955). The PRINTER ONLY bit remains set for the duration of the print cycle, and causes the main program to avoid testing whether I/O or keyboard requests are pending. From this point, the sequence of instructions followed is the same as if the task on entry was 1.

The basic constraints on the print positioning timing are dictated by the need to carry out the overall positioning within approximately 0.25 seconds as determined by the time it takes an envelope moving down the feed path to reach a position under the print head. Every command to a stepper lasts 6.4 or 12.6 milliseconds (if not able to move to the next position within 6.4 milliseconds). In order to result in a more even power drain, stepper commands are staggered at 1.6 millisecond intervals. The TIMER WAKEUP is turned off, a 1.6 millisecond timer is restarted and the initial command is sent (block 957). The initial command is in fact the last command that was sent to the particular stepper to make sure that the stepper is where it is thought to be, any possible difference being due to mechanical vibration and the like.

Tasks 1-3 are shown more specifically in Fig. 20b. Still in task 1, the routine tests whether all four steppers have been serviced (logical branch 960). If not, PRSTA is adjusted to prepare for the next stepper in sequence (block 962). The routine then returns to the main program after doing an initialization check which does the following: checks that in the initialization mode (logical branch 965) and that initialization is completed (logical branch 967). If it is, the steppers are turned off and the TIMER, TAPE/ENVELOPE, and HOME/ CLUTCH WAKEUP bits are cleared (block 970) before returning to the main program.

The routine then loops so that on subsequent passes the initial command is sent to the remaining steppers (block 957), and when it is determined at logical branch 960 that all four steppers have been done, the task is set to 2 (block 972). The bits in PRSTA are set for the next stepper in the cycle (which at this point is the first one), and after the initialization check, the routine returns to the main program.

On the next pass through the print routine, the task is 2. After branching at block 952, the routine turns off the TIMER WAKEUP and restarts the 1.6 millisecond timer. The routine tests whether the stepper is at its next position (logical branch 975), and if it has not reached its next position within the 6.4 milliseconds, a bit is set (block 978), assuming that this was not the second try as determined at logical branch 980. If it was a second try, then it is checked to see if doing initialization (logical branch 981). If yes, a jump is made out of the print routine to completely restart the initialization of the printer routine from stepping to '9999' (block 982). Else, a soft fault is done (block 983).

Assuming the stepper has reached its next position, the routine branches according to whether task 2 or 3 is being executed (logical branch 984). If all four steppers have been serviced (logical branch 985), the routine checks whether they have all reached their next position (logical branch 988). For task 2 the next position is the final position.

If all the steppers have reached their next position, the routine branches (according to whether task 2 or task 3 is being executed (logical branch 990). On task 2 the digits of the keyboard register are copied to the least significant nibbles of the respective STEPIN registers (block 992). The routine then determines the direction of stepping (block 995). The task is then set to 3 (block 1000) and the routine returns to the main program.

On a subsequent entry during task 3, the routine branches at logical branch 982 in order to check whether the

final position has been reached (logical branch 1002). If the final position has not been reached, the TIMER WAKEUP is turned off, the 1.6 millisecond timer is restarted, and the next command is sent out (block 1005). Once the final position has been reached, the routine tests whether all steppers have been serviced (branch 985), and branches as in task 2. Following logical branch 990, the timer is turned off, the solenoid is turned on and the task is set to 4 (block 1007) before doing the initialization check and returning to the main program.

On subsequent passes through the routine, it is ascertained at logical branch 940 that positioning has been completed, and tasks 4-7 are carried out. These tasks are shown specifically in Fig. 20c.

The routine first checks whether a TIMER WAKEUP has occurred (logical branch 1010). As will be discussed below, on subsequent tasks, this test may be true, but on task 4 it will not. The routine checks whether the TAPE/ENVELOPE WAKEUP bit is set (logical branch 1012) and if it has, the routine checks whether another envelope is on the way (logical branch 1015). If so, the task is set to 6 or 7, depending upon whether it was 4 or 5 (block 1017). The TAPE/ENVELOPE WAKEUP is turned off prior to a return to the main program (block 1020). Assuming that neither the TIMER WAKEUP nor the TAPE/ENVELOPE WAKEUP had been set, the only cause for entry into the print routine is that the HOME/CLUTCH WAKEUP bit had been set. This WAKEUP bit is turned off (block 1022) and a test is made whether the LEFT HOME signal indicates that the print head has left its home position (logical branch 1025). Assuming the print head has not left its home position, the routine checks whether the CLUTCH signal is present (logical branch 1027). If the task is 5 or 7, the routine returns to the main program at logical branch 1030. Otherwise, a 100 millisecond timing interval is set (block 1032). The purpose of this timing interval is to enforce the requirement that the LEFT HOME signal is received within 100 milliseconds of the CLUTCH signal. A failure of this to occur evidences a loss of home sensor 300, which loss

would leave the meter with no redundancy in the event of a failure in the base. This failure manifests itself by the foreground's setting the TIMER WAKEUP. This will be detected on subsequent entry at logical branch 1010, and is handled by causing the meter to fault with hexadecimal code 2 (block 1033). After setting the 100 millisecond timer, which occurs on task 4 or 6, the ascending and descending registers (AR1, AR2, DR1 and DR2) and the BATCH and COUNT registers in the BAMS are updated and the task is set to 5 or 7 prior to a return to the main program (block 1035).

If, at logical branch 1025, it is ascertained that the print head has left home, the timer is turned off (block 1037). On task 5 or 7 (as determined at logical branch 1038) the routine returns to the main program. Otherwise, the BAM and task updates (block 1035) are made before returning to the main program.

If it is determined at logical branches 1025 and 1027 that the CLUTCH signal and the LEFT HOME signal are both absent, the routine returns to the main program on tasks 4 or 6 (logical branch 1040). If the task is 5 or 7, this state indicates that the print head has returned to its home position. The routine tests whether the task is 7 (logical branch 1045), and turns off the steppers and solenoid (block 1047), if the task is 5, while they are left on if the task is 7. If task 5, the task is set to 0 (block 1052) while if the task is 7, the task is set to 4 (block 1055). New updated values of the registers are stored in the temporary registers in the BAMs (block 1057). If the task is 5, a check is made whether a single tape was to be printed (logical branch 1060). If a single tape was to be printed, the keyboard register is cleared and the routine calls the subroutine CHKTOT (block 1062) to make sure that the new registers adhere to the basic arithmetic constraint. Where envelopes or multiple tapes are to be printed, the temporary registers and keyboard register are set up to so that a subsequent print cycle can occur without requiring a new keyboard entry. A single tape (not repeat mode), on the other hand, once printed, causes the keyboard register to be cleared.

Prior to returning to the main program, the available funds are checked (logical branch 1065) and if it is found that insufficient funds remain to print another tape or envelope of the keyboard register value, "Add $" light 22 is lit, the steppers and solenoid are turned off, and the task is set to 0 (block 1067). If the task has been set to 4, indicating that a new print cycle is to occur, the program returns to the main program while if the task is 0 the PRINTER ONLY bit is turned off to enable the keyboard and I/O (block 1070).

Figs. 21a and 21b, taken together, form a flow chart of the keyboard routine. Generally, the keyboard routine discovers what keyboard key or keys have been pressed and takes action accordingly, as for examply by updating the keyboard register or various BAM registers. This routine is entered when the background dispatcher loop detects the KEYBOARD WAKEUP bit to have been set, which as discussed above occurs when the foreground routine detects a change to REG89 or REG07. In addition to keyboard key status, REG89 contains the status of switches 27 (fractional cents entry) and 30 (POMODE). The flow through this routine is controlled by the least significant nibble of KEYSTA, designated CC for current character and by a 1-bit flag, designated F0. Usually CC is set equal to the current keyboard switch being pressed, but it may assume values outside the range 0-9 to indicate special circumstances. In particular, if no keys are depressed, CC = 10; if keys "7", "8", and "9" are all pressed to signify that all the display segments are to be energized, CC = 12; and if the keyboard register KB is full, CC = 14. As discussed above, in the "mail room" mode, four characters may be entered before the keyboard register KB is considered full (three if the meter 5 only contains three steppers). In the "post office" mode, the entry capability expands to the number of characters in the maximum permissible value (e.g. seven characters if the maximum value allowed is $99,999.99). The determination whether KB is full is on the basis of the keyboard register character counter KBCC which is the tenth nibble of KB.

Before describing the precise operation and sequence of the keyboard routine, it is helpful to consider the various ways in which exit from the keyboard routine occurs. Generally, a return to the main program (background dispatcher loop) occurs via one of four exit points 1100, 1102, 1105, and 1107, designated A, B, C, and D, respectively. Upon branching to exit point A, the routine tests whether the printer is still busy (logical branch 1110), returning to the main program if it is and turning off the I/O and KEYBOARD WAKEUP bits and the PRINTER ONLY bit if it is not, prior to returning to the main program (block 1112). Upon branching to exit point B, the routine sets CC = 10 (block 1115) and returns via exit point A. Upon branching to exit point C, the routine clears the keyboard register and sets up a register request (block 1117), displays the keyboard register if the meter is in the "mail room" mode (block 1120), and returns via exit point B. Upon a branch to exit point D, the routine lights "Add $" lamp 22 clears the keyboard register, sets CC = 14 (block 1122), and returns via exit point A.

Upon entry into the keyboard routine, the foreground is immediately disabled and the routine initialized (block 1125). For definiteness, consider first the case where a single numeric key has been pressed. Prior to entry into the routine, CC will have been initialized to 10 (or left set at 10 from the last time that all keys had been released). The routine tests whether CC = 12 (block 1127), and upon finding it not equal to 12, sets flag F0 = 1 only if CC = 14 (block 1130). On this entry, F0 will not be set equal to 1. The routine then tests whether keys "7", "8", and "9" are all depressed (logical branch 1132), and upon finding such not to be the case, the routine undertakes to discover which key has been depressed (software vector 1135). This test is made according to a specific hierarchy wherein the keys are tested in the order "7", "8", "9", "4", "5", "6", "1", "2", "3", CLEAR BATCH, "0", and CLEAR KB. This particular order is appropriate for a standard calculator configured keyboard and in some measure avoids the problem

wherein a right handed person, in addition to the intended key, depresses the key immediately to the right or below. Assuming a numeric key to have been discovered as the first key to be depressed, the routine tests whether the keyboard register is full, i.e. whether F0 = 1 (logical branch 1137).

If the keyboard register is full, the routine returns to the main program via exit point A. If it is not full, a test is made whether CC = 10 (logical branch 1140) and returns via exit point A if CC is not equal to 10 since this would indicate another key to still be pressed. If CC = 10, the routine sets CC equal to the numeric character found (block 1142). The routine then increments KBCC and updates the keyboard register by left shifting the current contents and entering the newly discovered character in the rightmost position (least significant) (block 1145). The precise nibble which is considered the rightmost or least significant position is nibble 0 if the meter is in the "mail room" mode, and is a fractional cents meter, and selector switch 27 is set for fractional cents entry. Otherwise, it is nibble 1 of the keyboard register. If with the additional character, the keyboard register is full, the routine sets CC = 14 (block 1147), and then sets up a register request (block 1150). The routine then tests whether the meter is in the post office mode (logical branch 1152) and returns via exit point B if it is. Otherwise, the temporary registers TDR and TAR in the BAMs are updated as described above (block 1155). The checked arithmetic algorithm is carried out to make sure that the updated TAR and TDR sum to the control total (logical branch 1157), and if not, the meter faults with a hexadecimal code 7 written into the BAM location reserved for that purpose (block 1160). Under normal (non-fault) circumstances the routine then displays the new keyboard register (block 1162), tests whether the updated decending register has become negative (logical branch 1165), and if not, returns via exit point A. If TDR has become negative, the routine returns via exit point D.

Assume that keys "7", "8", and "9" are all depressed and that the current pass through the routine is in

response to the last one of these keys to have been depressed. At logical branch 1132, the routine branches differently from the above described sequence, sets up the segment codes so that all segments in the displays will be illuminated, and clears the keyboard register (block 1167). Then the routine sets CC = 12 (block 1170) prior to a return via exit point A. The next pass through the keyboard routine finds CC = 12 at logical branch 1127, whereupon the routine tests whether all of keys "7", "8", and "9" have been released (logical branch 1172). If not all of these three keys have been released, the routine returns via exit point A, but if all of the keys have been released, the routine returns via exit point C in order to clear the keyboard register, turn off the display of all segments, and set CC = 10.

If, at software vector 1135, it is determined that CLEAR KB is the only key to be depressed, the routine exits via exit point C. If it is found that no key is currently being depressed, the routine tests whether the entry into the routine was due to a new change into the "post office" mode (logical branch 1175) and returns via exit point C if it was. Otherwise, the routine returns via exit point B if CC is found equal to 12 or 14, and by exit point A otherwise (logical branch 1177).

It will be recalled that action in response to the CLEAR BATCH key occurs when the key is released, and may require the CLEAR KB key to have been depressed in the interim. In the event that the CLEAR BATCH key is discovered to have been depressed, the routine initially sets F0 = 0 (block 1180) and enables the foreground (block 1182) in preparation for waiting until the key is released. The routine then executes a wait loop, 1185, during which it tests whether F0 has been set to 1 (logical branch 1187). Initially, F0 will be 0 and the routine sets F0 = 1 when and if it finds the CLEAR KB switch to have been depressed (block 1190)). The routine then tests whether the CLEAR BATCH key has been released (logical branch 1192), and if not, loops back to logical branch 1187. When the clear batch key is

released, as discovered at logical branch 1192, the foreground is again disabled (block 1195).

The routine tests whether the meter is in the "post office" mode (logical branch 1197), and if so, adjusts temporary register TDR up or down by the keyboard register contents, depending on whether the CLEAR KB key had been depressed during the time that the CLEAR BATCH key had been pressed (block 1200). The routine then tests whether the descending register value would be in the permissible range (logical branch 1202). If not, the routine returns via exit point D in order to display the "Add $" lamp. If the descending register value is a permissible one, the routine adjusts the control total temporary register in the same direction as it had adjusted the descending register and updates permanent registers DR and TOTAL (block 1205) prior to returning via exit point C. If at logical branch 1197, the routine finds that the meter is in the "mail room" mode, the routine tests whether F0 = 1 (logical branch 1207) to determine whether the CLEAR KB key had been depressed during the time that the CLEAR BATCH key was depressed. If so, the routine clears the keyboard and batch registers (block 1210) prior to a return via exit point C. Otherwise, the routine returns via exit point B and in effect ignores depression and release of the CLEAR BATCH key.

The register display routine is entered in response to a REGISTER DISPLAY WAKEUP, and is responsible for determining which register is to be displayed and displaying it. The routine determines which display to display the information on, and takes care of decimal point placement and blanking leading zeros. The register display routine is straightforward and will not be described in further detail.

The I/O routine is entered in response to an I/O WAKEUP and has the function of receiving signals from outside the meter in order to carry out those functions that would be carried out in response to various combinations of keyboard key depressions and selector switch positions. The I/O routine will not be described in detail.

In summary, it can be seen that the present invention provides a microcomputerized postage meter having a high degree of security and fault tolerance so that critical register data is preserved under almost any conceivable failure condition. Once a failure has occurred, the meter recognizes its own lack of redundancy, and hence susceptibility to losing the data, and responds to this condition by activating internal circuitry for disabling the meter and preventing further operation until the meter is reset at the factory. Although disabled, the data in the registers is accessible for possible diagnostic purposes.

While the above description provides a full and complete disclosure of the preferred embodiment of the invention, various modifications, alternate constructions, and equivalents may be employed without departing from the true spirit and scope of the invention. For example, while the circuitry for inhibiting meter functioning upon detection of a fault condition includes a specific type of flip-flop, other flip-flop types or settable-resettable circuit elements could be adapted for use with the present invention. Similarly, while the use of stepper motors and mechanical verification contacts represents a preferred and relatively economical way to accomplish print wheel setting, other actuating mechanisms will be readily apparent to those of ordinary skill in the art. Also, the particular keyboard sequences and responses, while representative of appropriate data management, can be varied so long as such variations are carried out in a consistent manner. Accordingly, the above description and illustration should not be construed as limiting the scope of the invention, which is defined by the appended claims.

A computer program for the invention is shown by was of example in appended Appendix 1.

IN THE CLAIMS:

1. In a microcomputerized postage meter having a microcomputer programmed for supervising the printing operation and for maintaining and verifying accounting information, a postage printing mechanism coupled to the microcomputer for printing postage in response to instructions from the microcomputer, and input means coupled to the microcomputer for communicating data to the microcomputer; the improvement comprising:

first and second independent non-volatile memory means coupled to the microcomputer, each non-volatile memory means including self-contained power supply means and a plurality of memory locations corresponding to accounting information;

the programmed microcomputer including means for storing an item of accounting information in corresponding locations of the first and second non-volatile memory means, for retrieving the contents of the corresponding locations, and for comparing the contents of the corresponding locations; and

means coupled to the microcomputer for deactivating the meter in response to a disagreement between the contents of the corresponding locations.

2. In a microcomputerized postage meter having a microcomputer programmed for supervising the printing operation and for maintaining and verifying accounting information, a postage printing mechanism coupled to the microcomputer for printing postage in response to instructions from the microcomputer, and input means coupled to the microcomputer for communicating data to the microcomputer; the improvement comprising:

non-volatile memory means coupled to the microcomputer including self-contained power supply means and a memory location corresponding to accounting information;

the programmed microcomputer including means for storing an item of accounting information in a location of the non-volatile memory means, and for retrieving the contents of the location;

means for retrieving the contents of the location while the microcomputer is in an inactive state with respect to the non-volatile memory means.

3. The invention of claim 2 wherein the inactive state is an unpowered state.

4. The invention of claim 2 wherein the inactive state is a state wherein the microcomputer is incapable of accessing the non-volatile memory means.

5. In a microcomputerized postage meter having a microcomputer, a postage printing mechanism coupled to the microcomputer for printing postage in response to instructions from the microcomputer, and input means coupled to the microcomputer for communicating data to the microcomputer; the improvement comprising:

first and second independent non-volatile memory means coupled to the microcomputer, each non-volatile memory means including self-contained power supply means and a plurality of memory locations;

the programmed microcomputer having means for storing a set of accounting information items, the values of which items have an internal relationship to one another irrespective of their individual values, in corresponding pluralities of locations in the first and second non-volatile memory means, for retrieving the contents of the corresponding pluralities of locations, and for comparing the contents of the corresponding pluralities of locations; and

means coupled to the microcomputer for deactivating the meter in response to a disagreement between the contents of the corresponding pluralities of locations;

such that the contents of the corresponding pluralities of locations permit an operator to determine which of the two disagreeing sets of values is correct by checking which of the two sets of values satisfies the internal relationship.

6. In a microcomputerized postage meter having a microcomputer operable according to a program for supervising the printing operation and for maintaining and verifying accounting information, postage printing means coupled to the microcomputer for printing postage in response to instructions from the microcomputer, and input means coupled to the microcomputer for communicating data to the microcomputer; the improvement comprising:

first and second independent non-volatile memory means coupled to the microcomputer;

monostable multivibrator means having an input and an output;

the multivibrator output capable of assuming first and second distinct logic levels;

the multivibrator means being operable to maintain the multivibrator output at the first level for a predetermined time in response to a signal;

means coupled to the multivibrator output for disabling the meter in response to the second logic level on the multivibrator output; and

enable signal generating means coupled to the microcomputer and to the multivibrator input for repeatedly generating an electrical signal at intervals less than said predetermined time during operation of the meter;

the programmed microcomputer having means for suppressing the enable signal generating means in response to the detection of a failure condition, and means for storing in the first and second non-volatile memory means a code representative of the type of failure detected such that the detection of a failure causes the multivibrator output to assume the second logic level, wherein the meter is disabled.

7. In a microcomputerized postage meter having a microcomputer, and input means coupled to the microcomputer for generating signals representative of the value of postage to be printed, the improvement comprising:

a print element having a plurality of serially disposed positions, each position representative of a value of postage to be printed;

stepping motor means coupled to the print element, the stepping motor means having a plurality of positions corresponding to the plurality of print element positions;

motor driving means coupled to the microcomputer for driving the motor means from a first of the plurality of positions to a second serially adjacent one of the plurality of positions; and

position indicator means coupled to the microcomputer for generating and communicating to the microcomputer an electrical signal representative of the position of the stepping motor;

the programmed microcomputer having means for activating the motor driving means for a fixed time interval, means for determining whether the stepping motor means has moved from the first position to the second position within the fixed time interval, and means for suspending operation of the meter in response to a determination that the stepping motor means has not moved from the first position to the second position within the fixed time interval.

8. In a microcomputerized postage meter having a microcomputer, a postage printing mechanism coupled to the microcomputer for printing postage in response to instructions from the microcomputer, and input means coupled to the microcomputer for communicating data to the microcomputer; the improvement comprising:

a non-volatile memory unit including first and second independent non-volatile memory means coupled to the microcomputer, each nonvolatile memory means including self-contained power supply means and a plurality of memory locations;

the programmed microcomputer having means for storing a set of accounting information items, the values of which items have an internal relationship to one another irrespective of their individual values, in corresponding pluralities of locations in the first and second nonvolatile memory means, means for storing the set of accounting information items in a third plurality of locations in the non-volatile memory unit, means for changing the values of at least one of the items stored in the third plurality of locations in response to a signal from the input means, the changed values maintaining the same internal relationship, and means for copying the changed values into the first and second pluralities of locations when the postage printing mechanism is activated in accordance with the signals from the input means.

9. In a microcomputerized postage meter having a microcomputer, a postage printing mechanism coupled to the microcomputer for printing postage in response to instructions from the microcomputer, and input means coupled to the microcomputer for communicating data to the microcomputer, the microcomputer and the printing mechanism being located in a secure housing, the improvement comprising:

non-volatile memory means coupled to the microcomputer, including self-contained power supply means and a plurality of memory locations for storing a set of accounting information items;

bistable multivibrator means including self-contained power supply means and having an output capable of assuming first and second distinct logic levels in response to first and second respective states of the bistable multivibrator means, the bistable multivibrator means, once in the second state, being resettable to the first state only by physical access to the interior of the secure housing;

means coupled to the microcomputer for causing the bistable multivibrator means to assume its second state in response to the detection of a failure condition;

means responsive to the logic level of the bistable multivibrator output for generating a system clear signal in response to the appearance of the second logic level on the bistable multivibrator means output;

power surveillance means for generating a power loss signal in response to a low power condition;

the means for generating the system clear signal being further responsive to the power loss signal for generating the system clear signal;

means responsive to the system clear signal for preventing data transmission to the non-volatile memory means; and

means responsive to the system clear signal for preventing activation of the printing mechanism.

10. The invention of claim 9 also comprising:

means responsive to the appearance of the second logic level on the bistable multivibrator output for generating a microcomputer clear signal;

the means for generating the microcomputer clear signal being further responsive to the power loss signal for generating the microcomputer clear signal; and

means responsive to the microcomputer clear signal for disabling the microcomputer.

11. The invention of claim 1 or 5 wherein the microcomputer and the printing mechanism are located in a secure housing, and wherein the means for deactivating the meter comprises:

bistable multivibrator means including self-contained power supply means and having an output capable of assuming first and second distinct logic levels in response to first and second respective states of the bistable multivibrator means, the bistable multivibrator means, once in the second state, being resettable to the first state only by physical access to the interior of the secure housing;

means coupled to the microcomputer for causing the bistable multivibrator means to assume its second state in response to the detection of a failure condition;

means responsive to the logic level of the bistable multivibrator output for generating a system clear signal in response to the appearance of the second logic level on the bistable multivibrator means output;

means responsive to the system clear signal for preventing data transmission between the microcomputer and the non-volatile memory means;

means responsive to the system clear signal for preventing activation of the printing mechanism;

means responsive to the appearance of the second logic level on the bistable multivibrator output for generating a microcomputer clear signal; and

means responsive to the microcomputer clear signal for disabling the microcomputer.

12. The invention of claim 11 also comprising:

power surveillance means for generating a power loss signal in response to a low power condition;

the means for generating the system clear signal being further responsive to the power loss signal for generating the system clear signal; and

the means for generating the microcomputer clear signal being further responsive to the power loss signal for generating the microcomputer clear signal.

13. The invention of claim 11 wherein the means for preventing data transmission between the microcomputer and the non-volatile memory means permits data to be read from the non-volatile memory means independent of the microcomputer.

0019515

APPENDIX 1

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V3.0        PAGE   1
FME 925A METER. 4-10-79

LOC  OBJ         SOURCE STATEMENT

                 ;*************************************************************
                 ;
                 ;
                 ;       SYMDEF--THIS CONTAINS ALL THE SYMBOL DEFINITIONS FOR
                 ;               THE FRIDEN ELECTRONIC POSTAL METER MICRO PROGRAM
                 ;
                 ;
                 ;*************************************************************
                         ;
                         ;FOLLOWING ARE THE SYMBOL DEFINITIONS FOR PORT 2 MANIPULATIONS
                         ;
                         ;NOTE THAT TO SUPPORT A MIXTURE OF THE PROMT-42, EPRM12, EPRM2,WRIBR,
                         ;AND SWRIBR ALL HAVE F (HEX) IN THE LSN IN ORDER TO READ BR
0040             SOLEN   EQU     040H    ;USED FOR ANL P2,# TO KEEP SOLENOID IN PRESENT STATE
004F             SOLNFF  EQU     SOLEN OR 0FH    ;FOR USE BY WRITE
002F             EPRM2   EQU     02FH    ;USED FOR ORL P2,# TO WRITE TO 2ND BRM
000F             EPRM12  EQU     00FH    ;USED FOR ORL P2,# TO READ THE TWO BRMS
001F             EPRM2   EQU     01FH    ;USED FOR ORL P2,# TO READ THE SECOND BRM
0030             DPRDS   EQU     030H    ;USED FOR ORL P2,# TO DISABLE PROG 1 & 2
                                         ;AND THE TWO BRMS
000F             WRIBR   EQU     00FH    ;USED FOR ORL P2,# TO WRITE TO BRM
007F             SWRIBR  EQU     07FH    ;USED FOR ANL P2,# TO STOP THE ABOVE WRITE
0010             PROG2   EQU     010H    ;USED FOR ORL P2,# TO WRITE TO DISP AND VERIFY AND
                                         ;TEST 8243
001F             PROG2F  EQU     PROG2 OR 0FH    ;FOR USE BY WRITE
0020             POS1    EQU     020H    ;USED FOR ORL P2,# TO POSITION STEPPERS
                         ;
                         ;FOLLOWING ARE SEVERAL MISCELLANEOUS SYMBOLS
                         ;
00FF             MINONE  EQU     0FFH    ;ALL ONES FOR ANL OR ORL
0007             MPST1   EQU     007H    ;ADDR FOR 74145 TO DO MPST1 (FIRE ONE SHOTS FOR
                                         ;FAULT FFS.
0032             SERIPH  EQU     032H    ;SERIPH CHAR FOR 4 CHAR DISPLAY
0003             ZEROSC  EQU     003H    ;ENCODE OF BCD '0' FOR 9 CHAR DISPLAY
000F             CLRMSN  EQU     00FH    ;FOR ANL X,# TO CLEAR MS NIBBLE
00F0             CLRLSN  EQU     0F0H    ;FOR ANL X,# TO CLEAR LS NIBBLE
00FB             TIMCNT  EQU     0FBH    ;# OF 80 MICROSEC INTERVALS FOR EACH
                                         ; TIMER INTERRUPT (5*80=400 MICROSECS)
0010             CHGDIR  EQU     010H    ;FOR XRL TO CHANGE DIR OF STEPPING
                         ;
                         ;FOLLOWING ARE SOME REGISTER DEFINITIONS (IN RAM)
                         ;
001A             R2FORG  EQU     01AH    ;ADDR OF R2' (FOREGROUND) FOR INDIR ACCESS
001F             R7FORG  EQU     01FH    ;ADDR OF R7' (FOREGROUND) FOR INDIR ACCESS
002A             REG07   EQU     02AH    ;ADDR OF INTERNAL REPRESENTATION OF SWITCHES 0-7
002C             REG89   EQU     02CH    ;ADDR OF INTERNAL REPRESENTATION OF SWITCHES 8,9,CLR
                                         ;KEYBD AND BATCH. P.O MODE.NORM/FRAC.FCENTS.
002E             REGSW   EQU     02EH    ;ADDR OF INTERNAL REPRESENTATION OF REG SEL SWITCHES.
                                         ;TAPE,ENV,REPEAT
002B             LSTC07  EQU     02BH    ;ADDR OF LAST TIME CHANGE OF REG07
002D             LSTC89  EQU     02DH    ;ADDR OF LAST TIME CHANGE OF REG89
002F             LSTCSW  EQU     02FH    ;ADDR OF LAST TIME CHANGE OF REGSW
0034             KEYSTA  EQU     034H    ;ADDR OF CURR STATUS OF KEYBOARD TASK. MSN = INFO
                                         ;ABOUT P.O , F/N. FCENTS. LSN = CURR KB CHAR.
```

BAD ORIGINAL

0019515


```
LOC  OBJ          SOURCE STATEMENT

0017              METYPE  EQU   017H  ;ADDR OF METER TYPE INFO. BIT 7 = 1 SAYS 4 STEPPER
                                      ;METER, =0 SAYS 3:  BIT 6 IS PCENTS BIT;  BIT 5 IS
                                      ;1 IF UPS METER,  LSN TELLS HOW MUCH
                                      ;P.O. CAN ENTER INTO PC
0036              PRCTR   EQU   036H  ;PRINT WAIT COUNTER   IF NE 0 IS COUNTED DOWN BY FOREGROUND
                                      ;AND WHEN GOES TO ZERO IS REPORTED TO MAIN ROUTINE
                                      ;THRU BASTR
0035              PRTSTA  EQU   035H  ;ADDR OF CURR STATUS OF PRINTER ROUTINE,
                                      ;MSB SAYS INITIALIZATION. NEXT 3 BITS TELL
                                      ;WHAT PHASE OF PRINTING. BITS 2&3 TELL IF
                                      ;TAPE OR ENV. AND REPEAT. AND BITS 0&1
                                      ;TELL WHICH STEPPER TO STEP IF NECESSARY.
0016              IOSTR   EQU   016H  ;ADDR OF CURR STATUS OF I/O ROUTINE
0037              IOPTR   EQU   037H  ;ADDR OF PTR USED BY I/O TO READ OR WRITE BAM
0029              SAVACC  EQU   029H  ;ADDR WHERE THE TWO INTERRUPT ROUTINES STORE THE
                                      ;BACKGROUND'S ACCUMULATOR
0002              BASTR   EQU   002H  ;ADDR (R2) OF BACKGROUND STATUS REG
0039              STEPIN  EQU   039H  ;ADDR OF 4 INDEXED BYTES GIVING INFO ABOUT
                                      ;THE STEPPERS.  LSN = FINAL DESIRED POSITION, MSN =
                                      ;PRESENT COMMAND TO THE STEPPER
003C              STEPTK  EQU   03CH  ;ADDR OF 4 INDEXED BYTES GIVING FURTHER INFO
                                      ;ABOUT THE STEPPERS.  MSN=STATUS INFO FOR EACH STEPPER.
                                      ;LSN = NEXT POSITION CODE EXPECTED
003E              STPTK2  EQU   03EH  ;ADDR OF STEPTK(2)
                  ;
                  ;THE FOLLOWING DEFINITIONS ARE TO SET AND CLEAR BITS IN BASTR
                  ;
0008              KEYTSK  EQU   008H  ;FOR ORL A,# TO SET KEYBD TASK TO ONE
0001              INHBEI  EQU   001H  ;FOR ORL A,# TO INHIBIT I/O AND KB TASKS
0010              SWTSK   EQU   010H  ;FOR ORL A,# TO SET REG SWITCH TASK TO ONE
FFEF              NSWTSK  EQU   NOT SWTSK      ;FOR ANL A,# FOR CLEAR ABOVE TASK
0020              PTITSK  EQU   020H  ;FOR ORL A,# TO SET PRINTER TIMED TASK TO ONE
FFDF              NPTTSK  EQU   NOT PTITSK     ;FOR ANL A,# FOR CLEAR ABOVE TASK
0004              IOTSK   EQU   004H  ;FOR ORL A,# TO SET I/O TASK TO ONE
FFFB              NIOTSK  EQU   NOT IOTSK      ;FOR ANL A,# TO CLEAR ABOVE TASK
FFF2              NKYIOP  EQU   NIOTSK AND NOT KEYTSK AND NOT INHIBI   ';FOR ANL #A AT
                                      ;END OF KEYBOARD TASK WHEN NOT PRINTING
0080              PFTSK   EQU   080H  ;SET UPON FINDING POWER LOSS COMING TRUE
0002              PHOCL   EQU   002H  ;SET UPON FINDING CHANGE IN HOME OR CLUTCH
FFFD              NPHOCL  EQU   NOT PHOCL      ;USED TO TURN OF PHOCL
0040              PTAEN   EQU   040H  ;SET UPON FINDING CHANGE IN TAPE OR ENVELOPE
FFBF              NPTAEN  EQU   NOT PTAEN      ;FOR TURNING OFF TAPE/ENV WAKEUP
FFBD              NPETHC  EQU   NPTAEN AND NPHOCL      ;USED TO TURN OFF PTAEN &/OR PHOCL
FF9F              NPETTI  EQU   NPTAEN AND NPTTSK ;USED TO TURN OFF PTAEN &/OR PTITSK
FFF3              NKBIOT  EQU   NOT KEYTSK AND NIOTSK   ;USED TO TURN OFF THESE 2 BITS
                  ;
                  ;FOLLOWING DEFINITIONS ARE FOR THE VARIOUS BAM REGISTERS
                  ;
0000              BDREG   EQU   000H  ;DECREMENTING REG(A)
0010              BAREG   EQU   010H  ;ASCENDING REG(B)
0020              BTOTAL  EQU   020H  ;CONTROL TOTAL(B)
002C              BCHECK  EQU   2CH   ;WHERE ALL 1'S STORED  TO VERIFY CORRECT
                                      ;READING OF BAM BY MAT
002D              BFAULT  EQU   02DH  ;WHERE FAULT INFO IS STORED
002E              BMAXPO  EQU   02EH  ;WHEN MAX P.O. MODE CODE IS STORED
```

BAD ORIGINAL

```
LOC  OBJ        LINE        SOURCE STATEMENT

002F            110 BMETYP  EQU     02FH    ;WHERE BITS ABOUT THE METER TYPE ARE STORED
0030            111 BBATL   EQU     030H    ;BATCH TOTAL(0)
0040            112 BBACT   EQU     040H    ;BATCH COUNT (0)
0050            113 BCOUNT  EQU     050H    ;COUNT OF PRINT CYCLES
0060            114 BKBREG  EQU     060H    ;KEYBOARD REGISTER (0) (BAM 1 ONLY)
006A            115 CKBCHR  EQU     06AH    ;ADDR OF # OF CHARS--IN BKBREG(A)
0070            116 BTDREG  EQU     070H    ;TEMP DREG(0) (BAM 1 ONLY)
0078            117 BTDRPS  EQU     078H    ;LOCATION USED BY KBTLG ROUTINE TO SEE IF
                118                         ;NEW VALUE OF DR WOULD BE TOO LARGE OR SMALL
0080            119 BTAREG  EQU     080H    ;TEMP AREG(0) (BAM 1 ONLY)
                120                         ;TEMP AREG(0) (BAM 2) USED TO STORE TEMP COPY OF TOTAL
0090            121 BTEMP   EQU     090H    ;TEMPORARY REGISTERS(0) (BOTH BAMS) USED TO DO
                122                         ;ADDS AND SUBS W/O MODIFYING OTHER REGISTERS
                123                         ;AFTER WHICH CHECKS ON EQUIVALENCY ARE DONE
00A3            124 BRGONE  EQU     0A3H    ;REGISTER WITH VALUE OF 1 FOR INCREMENTING COUNT
00A9            125 BRGTHO  EQU     0A9H    ;REGISTER WITH VALUE OF 1,000 FOR INCR BATCH CT
00B0            126 BTTDR   EQU     0B0H    ;START OF TEMP REGISTERS FOR LAST CHANGE SAVE
                127         ;
                128         ;THE FOLLOWING DEFINITIONS ARE FOR MOVD P5,A
                129         ;TO SELECT THE PROPER 74150 DEMULTIPLEXER INPUTS TO TEST ON T1
                130         ;
000F            131 DMPCLU  EQU     00FH    ;DEMPXR ADDR FOR CLUTCH
0000            132 DMPRIN  EQU     000H    ;DEMPXR ADDR FOR KEYBOARD INPUTS
0001            133 DMPKIN  EQU     001H    ;DEMPXR ADDR FOR REG SELECT SW INPUT
0005            134 DMPTAP  EQU     005H    ;DEMPXR ADDR FOR TAPE SIGNAL INPUT
                135         ;
                136         ;
                137         ;THE FOLLOWING DEFINITIONS ARE FOR THE PRINT ROUTINE
                138         ;
0080            139 STPM1   EQU     080H    ;USED TO TELL NEXT EXPECTED POSITION IS A 1/2
                140                         ;POSITION (RETURNS -1)
FF7F            141 NSTPM1  EQU     NOT STPM1       ;USED TO NEGATE THE ABOVE
0020            142 STPDON  EQU     020H    ;TELLS STEPPING MOTOR MOVED TO A FULL POSITION
0060            143 POSDON  EQU     060H    ;STEPPER MOTOR POSITIONED TO FINAL POSITION
FF9F            144 NDONEP  EQU     NOT STPDON AND NOT POSDON       ;STEPPER STARTING TO NEW POS
0004            145 STINTV  EQU     004H    ;# OF TIME INTERVALS FOR EACH 1/4  PRINT CYCLE,
                146                         ;400 MICRO SECS * 5 SAYS EACH STEP TAKES 2 MILLISECS
0040            147 STPTRY  EQU     040H    ;SET TO INDICATE ALREADY TRIED TO POSITION 1
                148                         ;PRINT CYCLE (ABOVE TIME).
                149         ;
                150         ;THE FOLLOWING DEFINITIONS ARE FOR THE I/O ROUTINE
                151         ;
0001            152 IORPTR  EQU     001H    ;COMMAND TO READY NEW PTR FROM UPI
0002            153 IORKB   EQU     002H    ;COMMAND TO READ NEW CHAR FOR KEYBOARD REG
0003            154 IORDB   EQU     003H    ;COMMAND TO READ BAM USING IOPTR
0004            155 IOWRBX  EQU     004H    ;COMMAND TO WRITE TO BAM USING IOPTR
0009            156 ENAIO   EQU     009H    ;PORT 6 DATA TO ENABLE I/O (EIO)
                157         ;
                158 ;***************************************************************
                159 ;
                160 ;
                161 ;       TIMINT -- S/R FOR TIMER INTERRUPT.  THOUGHT OF AS A FOREGROUND TASK
                162 ;               , THIS DISPLAYS THE TWO DISPLAYS, DEBOUNCES THE KEYBOARD
                163 ;               SWITCHES, AS WELL AS THE REGISTER SELECT SWITCHES,
                164 ;               THE SIGNALS:TAPE, ENVELOPE,AND REPEAT, READS THE SWITCHES
```

BAD ORIGINAL

0019515


```
LOC  OBJ        SOURCE STATEMENT

                ;       [illegible] P.O. MODE, AND THE INTERRUPTS (I/O). IT ALSO
                ;       DOES COUNTING FOR THE PRINTER ROUTINE IF NECESSARY. IT
                ;       COMMUNICATES WITH THE BACKGROUND TASKS THRU BAKSTA (R2), WHICH
                ;       THE BACKGROUND TASK MAIN USES TO START VARIOUS TASKS. IT
                ;       ALSO SETS UP VARIOUS REGISTERS FOR THE BACKGROUND TO READ
                ;       TO TELL WHAT IS THE PRESENT STATE OF THE OUTSIDE WORLD
                ;       R0 HOLDS THE CURRENT REGS. R1 IS USED FOR OTHER & DUTIES.
                ;       F1 IS USED. THERE FORE ONLY ROUTINES WHICH DISABLE TIMINT
                ;       CAN USE F1. TIMINT TAKES 115 MICROSECS EVERY 400 WHEN
                ;       DISABLED. HENCE ALL BACKGROUND TASKS TAKE 400/285 LONGER
                ;       THAN ACTUAL INSTRUCTION TIME
                ;
                ;
                ;***********************************************************************
                ;
                ;THIS IS START OF FRIDEN ELETRONIC POSTAL METER MICROCODE
                ;
0000                    ORG     0
0000 [illegible]        INITZE: JMP     INIT            ;COME HERE WITH R/P CLEAR
0003                    ORG     3
0003 [illegible]        SEL     RB1             ;FIRST PART OF NORINT
0004 [illegible]        MOV     R6,A            ;SAVE ACCUM
0005 [illegible]        JMP     NORINT          ;COME HERE WITH NORMAL INTERRUPT
0007                    ORG     7
                ;
0007 [illegible]        TIMINT: SEL     RB1             ;USES RB1 REGISTERS
0008 [illegible]        MOV     R7,A            ;SAVE BACKGROUND ACCUM IN REG 3FH
0009 [illegible]        MOV     A,#TIMOUT       ;REINITIALIZE THE COUNTER
000B [illegible]        MOV     T,A
000C [illegible]        MOV     R0,#PRTTR       ;DO STUFF FOR PRINTER TIMER, SET IT'S
000E [illegible]        MOV     A,@R1           ;ADDRESS AND FETCH
000F [illegible]        JZ      TMINT1          ;IF ZERO NOTHING TO DO
0011 [illegible]        DEC     A               ;ELSE DECR AND PUT BACK
0012 A1         MOV     @R1,A
0013 [illegible]        JNZ     TMINT1          ;IF NOT ZERO NO MORE TO DO
0015 [illegible]        MOV     R1,#BAKSTA      ;IT'S JUST BECAME 0 LET BAKSTA
0017 [illegible]        MOV     A,@R1
0018 [illegible]        ORL     A,#PTITSK       ;KNOW
001A [illegible]        MOV     @R1,A
001B [illegible]        TMINT1: IN      A,P2            ;IN CASE NODISP
001C [illegible]        MOV     R4,A
001D [illegible]        ANL     P2,#SGLEN       ;SET UP PORT 2 FOR FOREGROUND
001F [illegible]        ORL     P2,#[illegible]
0021 [illegible]        MOV     A,#002H         ;SEE IF RAM WRITE SIGNAL TRUE
0023 [illegible]        MOVD    P5,A            ;IF SO FAULT
0024 [illegible]        JT1     $
0026 [illegible]        TMINT[illegible] MOV     A,R2            ;IF MSB OF R2 = 1, NO DISPLAY
0027 [illegible]        JB7     NODISP
0029 A9         MOV     R1,A            ;SAVE IN R1 FOR LATER
002A 85         CLR     F0
002B 95         CPL     F0              ;F0 IS USED TO TELL IF DOING 4CHAR
002C [illegible]        JB4     TMINT2          ;(=1) OR 9CHAR DISPLAY (=0)
002E 85         CLR     F0              ;IF 4 CHAR R2 (A) = 3XH. IF 9 CHAR
                                ;R2 (ACCUM) = 2XH
002F [illegible]        ANL     A,#CLRMSK       ;TO TEST FOR CHAR COUNT
```

BAD ORIGINAL

```
LOC  OBJ        LINE        SOURCE STATEMENT

0031 C6CE       220         JZ      TIMIN2          ;IF ZERO DOING 1ST CHAR. JMP
0033 03F8       221         ADD     A,#0F8H         ;SEE IF DOING 9TH CHAR (8+F8=0)
0035 9641       222         JNZ     TIMIN1          ;NO, JMP
0037 0433       223         JMP     TIMIN2          ;AND JUMP
                224         ;
                225         ;THIS IS 40CHAR DISPLAY. SEE IF 1ST CHAR OF OTHER
                226         ;
0039 53C0       227 TMINT2: ANL     A,#CLRMSK       ;TO TEST FOR CHAR COUNT
003B 9645       228         JNZ     TIMIN5          ;LSN R2 IN A. JMP IF NOT ZERO
003D CA         229         DEC     R2              ;YES, DEC R2 (COUNTER FOR #
                230                                 ;OF TIMES TO GO THRU 40CHAR DISPLAY
                231                                 ;WHEN 0, LAST TIME BEFORE DOING 9 CHAR
                232                                 ;AGAIN.
003E F0         233 TIMIN2: MOV     A,@R0           ;GET R4 (CURRENT M/P REPRESENTATION
                234                                 ;OF REGXX DECIDED EARLIER). THEN R5
003F AC         235         MOV     R4,A
0040 18         236         INC     R0              ;(LAST TIME CHANGE REGISTER FOR REGXX)
0041 F0         237         MOV     A,@R0
0042 AD         238         MOV     R5,A
0043 BA00       239         MOV     R2,#0           ;AND CLEAR R2 (THIS TIME CHANGE REG)
0045 FA         240         MOV     A,R2            ;GET A READY
0046 B6??       241         JF0     TIMIN5          ;IF F0=1, DOING 4 CHAR
                242         ;
                243         ;TIMIN1 DOES DISPLAY OF 8 CHAR AND PRELIMINARY DEBOUNCING OF
                244         ;KEYBOARD SWITCHES 0-9
                245         ;
0048 FA         246 TIMIN1: MOV     A,R2            ;GET R2 AGAIN
0049 89FF       247         ORL     P1,#MINONE      ;MAKE P1 = -1 TO TURN OFF DISPLAY
004B 3F         248         MOVD    P7,A            ;A HAS CHAR. LSN IS ADDR FOR 74145
004C F1         249         MOV     A,@R1           ;R1 HAS PTR TO CHAR (THIS AND ABOVE
                250                                 ;ARE COPIES OF R2)
004D 39         251         OUTL    P1,A            ;SEND CODED CHAR TO P1 FOR DISPLAY
004E 27         252         CLR     A               ;GET READY TO READ PIN (T1) (=0)
004F 3D         253         MOVD    P5,A            ;SEND TO 8243
0050 5452       254         CALL    FIRDY           ;DOES PRELIMINARY SW DEBOUNCING
0052 FA         255         MOV     A,R2            ;GET CHAR COUNT TO SEE IF DONE WITH
                256                                 ;PHASE 07 OR 8?
0053 03D9       257         ADD     A,#0D9H         ;27H + 0D9H = 0. IF DONE WITH CHAR '7',
0055 C6??       258         JZ      DONE11          ;JUMP
0057 03FE       259         ADD     A,#0FEH         ;29H + 0D9H + 0FEH = 0 IF DONE
0059 C6??       260         JZ      DONE3           ;WITH CHAR '9', JUMP
005B 04??       261         JMP     TIMFIN          ;ELSE JUMP
                262                                 ;CALL DONE WHICH DOES FINAL
005D 74??       263 DONE11: CALL    DONE            ;DEBOUNCE STUFF
005F FF         264         MOV     A,R7            ;IF CHANGE F1=1. SAVE INFO IN R7(7)
0060 76??       265         JF1     TIN000
0062 537F       266         ANL     A,#07FH         ;NO CHANGE IN REG07
0064 AF         267 TIN000: MOV     R7,A            ;THIS IS LOOKED AT IN DONE3
0065 B8??       268         MOV     R0,#REG29       ;GET READY FOR NEXT PHASE OF DEBOUNCING
0067 04??       269         JMP     TIMFIN          ;JMP TIMFIN
                270         ;
                271         ;TIMIN5 IS LIKE TIMIN1 EXCEPT FOR 4 CHAR DISPLAY
                272         ;
0069 5900       273 TIMIN5: ANL     P1,#000         ;MAKE P1=0 TO BLANK DISPLAY
006B 3F         274         MOVD    P6,A            ;ADD HAS R2. LSN NIBBLE HAS ADDR FOR
```

BAD ORIGINAL

```
LOC  OBJ        SOURCE STATEMENT

[illegible]               [illegible]  [illegible]     ;[illegible] R1 ARE R2, POINTER TO ENCODED
[illegible]               [illegible]  R1,A            ;CHARACTER
[illegible]               MOV   [illegible]     ;GET [illegible] TO [illegible]
[illegible]               MOV   [illegible]
[illegible]               CALL  [illegible]     ;[illegible] PRELIM [illegible] DEBOUNCING
[illegible]               MOV   A,R2            ;ARE WE DOING [illegible]
[illegible]               [illegible]  A
[illegible]               JB2   [illegible]     ;NO, JMP
[illegible]               MOV   A,R2            ;[illegible] TO DO [illegible] IF HAVE GONE THRU
                                                ;[illegible] DISPLAY ENOUGH TIMES
[illegible]               MOV   R2,[illegible]  ;GET [illegible] IN CASE REPEATING 4
                                                ;CHAR DISPLAY
[illegible]               DEC   R0              ;IN CASE JMP
[illegible]               JNC   [illegible]     ;NOT YET, JMP
[illegible]               INC   R0              ;DIDN'T JMP GET BACK TO CORRECT
[illegible]               JMP   DONES           ;VALUE FOR DONE
                     ;
                     ;
                     ;DONE--COME HERE WHEN DONE WITH [illegible] CHAR DISPLAY. NEED TO DO CLR
                     ;[illegible] AND [illegible], AS WELL AS P.O. [illegible] AND
                     ;[illegible] ALSO DO EXTINTS CLUTCH AND HOME.
                     ;
[illegible]  DONE:        [illegible]  A          ;WANT TO TURN OFF [illegible] SWITCH DRIVE
[illegible]               [illegible]  [illegible]
[illegible]               MOV   [illegible]     ;[illegible] COUNTER FOR 4 CHAR
                                                ;DISPLAYS TEMPORARILY [illegible] LOOP CTR
[illegible]               MOV   [illegible]     ;[illegible] POINTER FOR [illegible]
[illegible]  [illegible]: [illegible]  A,R2        ;THIS LOOP DOES PRELIM DEBOUNCING
[illegible]               INC   R2              ;GET [illegible] FOR NEXT PASS
[illegible]               [illegible]  [illegible]   ;[illegible] CLR [illegible] CLR [illegible]
[illegible]               CALL  [illegible]
[illegible]               DJNZ  [illegible]     ;DONE YET?
[illegible]               MOV   [illegible]     ;YES [illegible] FOR ANOTHER LOOP
                                                ;R2 = 22H TO [illegible] P.O. [illegible]
[illegible]  [illegible]: CALL  [illegible]     ;AND THEN [illegible] MODE.
[illegible]               DJNZ  [illegible]     ;DONE YET?
[illegible]               CALL  [illegible]     ;YES, [illegible] R4 & R5
[illegible]               CALL  [illegible]
[illegible]               CALL  [illegible]     ;DO FINAL DEBOUNCING. R1 = 1 SAYS
                                                ;CHANGE (USED LATER)
                     
                     ;NEED TO [illegible] REG[illegible](7) = P.O. MODE + /[illegible] + [illegible]
                     ;I.E. = REG[illegible](4) + /REG[illegible] + REG[illegible](5)
                     ;
[illegible]               [illegible]  A,[illegible]   ;MAKE A [illegible] IF BIT 7 = 1
[illegible]               JB4   [illegible]     ;IT IS IF P.O. MODE
[illegible]               JB5   [illegible]     ;OR IT IS IF [illegible]
[illegible]               JB6   [illegible]     ;BUT NOT IF [illegible] = [illegible]
[illegible]               JMP   [illegible]     ;ELSE IT IS
[illegible]  [illegible]: ANL   A,#07FH         ;MAKE REG[illegible](7) = 0
[illegible]  [illegible]: MOV   [illegible],A   ;STORE IN REG[illegible]
[illegible]               MOV   A,R7            ;R7(7) HAS 1 IF CHANGE IN REG87
[illegible]               MOV   R[illegible],A  ;[illegible] IS FREE FROM DEBOUNCE STUFF, AND
                                                ;NEED TO SAVE R7 FOR LATER
[illegible]               JB7   [illegible]     ;JUMP IF CHANGE OCCURED
```

BAD ORIGINAL

```
LOC  OBJ       LINE      SOURCE STATEMENT

00AF 7609           JF1     DONE29      ;ELSE IF CHANGE IN REG29 JUMP
00AC BFC0      DONE37: MOV  R7,#0C0H    ;CLEAR OUT 1ST 6 BITS OF R7
                                        ;WHERE THE EXTINTS, HOME AND CLUTCH
                                        ;  002, ANY TEST
                                        ;WILL BE STORED  R7(6) = A FLAG
                                        ;TO TELL WHEN DONE ALL AND R7(7)
                                        ;WILL BE USED LATER DURING DONE1
                                        ;AND DONE3,R2 IS SET UP TO 29.
00AE 1A        DONE33: INC  R2
00AF FA             MOV     A,R2
00B0 3D             MOVD    P5,A        ;MOVE TO 74150 TO SET UP T1
00B1 FF             MOV     A,R7        ;GET R7
00B2 46B5           JNT1    DONE333     ;DON'T SET BIT UNLESS T1=1
00B4 17             INC     A
00B5 77        DONE333: RR  A           ;ROTATE FOR NEXT BIT
00B6 12BB           JB0     DONE34      ;IF FIND BIT A READY SET, DONE
00B8 AF             MOV     R7,A        ;ELSE MOV R7 TO A AND RELOOP
00B9 04AE           JMP     DONE33
00BB 77        DONE34: RR   A           ;DO 2 MORE ROTATES TO ALIGN
00BC 77             RR      A
00BD AF             MOV     R7,A        ;PUT BACK IN R7
00BE 32D2           JB1     DONE36      ;IF BIT 1 (EXTINT1) = 1, NEED TO
                                        ;MODIFY BAKSTA
00C0 FF        DONE35: MOV  A,R7        ;NEED TO SEE IF SEMI-DEBOUNCED
00C1 DE             XRL     A,R6        ; CHG TO CLUTCH OR HOME, OLD R7
00C2 5330           ANL     A,#030H     ;SAVED IN R6
00C4 C6CA           JZ      DONE3F      ;JUMP IF NO CHANGE TO HOME OR CLUTCH
00C6 2302           MOV     A,#PHOCL    ;SET BIT IN BAKSTA
00C8 74CE           CALL    BAKREQ
00CA BA30      DONE3F: MOV  R2,#030H    ;MAKE CHAR POINTER READY TO DO 4CHAR
00CC B82E           MOV     R0,#REG5H   ;GET R0 READY FOR DEBOUNCE OF REG5H5
00CE BB04           MOV     R3,#004H    ;AND PUT IT'S # OF TIMES IN R3
00D0 04EB           JMP     TIMRET
00D2 2304      DONE36: MOV  A,#IOTSK    ;HAVE I/O TASK TO SET UP IN BAKSTA
00D4 74CE           CALL    BAKREQ
00D6 04C0           JMP     DONE35
               DONE39:;     JB2     DONE38      ;DO K/B ONLY IF .00 + .000
                                        ;JUMP IF .00
               ;    MOV     A,R4        ;SEE IF .000
               ;    CPL     A           ;FOR EASE OF TEST
               ;    JB5     DONE37      ;NO K/B IF JUMP
00D8 2308      DONE38: MOV  A,#KEYTSK   ;CHANGE BAKSTA TO DO KEYBD TASK
00DA 74CE           CALL    BAKREQ
00DC 04AC           JMP     DONE37
                    ;
                    ;ENDS OF TIMINT ROUTINE
                    ;NODISP IS FOR USE BY TIMINT TO KEEP THE ONE SHOTS FOR THE FAULT FFS
                    ;GOING  IF THERE IS NO DISPLAY (CHAR
                    ;POINTER NEGATIVE)
                    ;ONESHT IS USED BY THE TIMINT ROUTINE DURING NORMAL DISPLAY
                    ;ACTION TO KEEP THE FAULT ONE SHOTS FIRING AT THE END OF DONES
                    ;TIMRET IS USED AT THE END OF DONE1 AND DONE3
                    ;TIMFIN IS USED AT THE END OF ALL OTHER PATHS THRU THIS ROUTINE
                    ;
                    ;COME FROM DONES WITH F0=1, F0=0
```

BAD ORIGINAL

```
LOC  OBJ         LINE        SOURCE STATEMENT

                  385         ;SAYS DOING NO DISPLAY, ONLY
                  386         ;PRINTER TIMER AND FAULT FF ONE SHOTS
                  387         ;
00DE 85           388 NODISP: CLR     F0              ;SAYS DOING NO DISP, ONLY PRCTR
                  389                                 ;AND ONE SHOTS
00DF 2307         390 ONESHT: MOV     A,#MPST1        ;GET READY TO FIRE 2 ONE SHOTS
00E1 3E           391         MOVD    P6,A
00E2 17           392         INC     A               ;TO MAKE CHANGE TO FAULT FFS
00E3 3E           393         MOVD    P6,A            ;ONE SHOTS
00E4 B6EB         394         JF0     TIMRET          ;JUMP IF AT END OF DONES
00E6 F9           395         MOV     A,R1            ;ELSE GET OLD PORT 2
00E7 3A           396         OUTL    P2,A
00E8 04EB         397         JMP     TIMRET
00EA 1A           398 TIMFIN: INC     R2              ;INC CHARACTER POINTER
00EB B929         399 TIMRET: MOV     R1,#SAVACC      ;INIT R1 FOR LEAVING AND ENTERING
00ED F1           400         MOV     A,@R1           ;INTERRUPT ROUTINES (USED TO
                  401                                 ;INDIRECTLY STORE AND FETCH BACKGROUND
                  402                                 ;ACCUMULATOR
00EE 93           403         RETR                    ;DOES SEL RB0
                  404 ;****************************************************************
                  405 ;
                  406 ;       WRTDSP--S/R WHICH READS FROM BAM TO RAM (ACTUALLY INTENDED
                  407 ;               FOR READING A BAM REGISTER AND CONVERTING CHARS TO
                  408 ;               7 SEG CODE FOR 4 CHAR DISPLAY AND WRITING TO RAM).
                  409 ;               SEVERAL PARAMETERS ARE PASSED IT.  R0 HAS WRITE ADDRESS
                  410 ;               (IN RAM) , R1 HAS BAM ADDR (BAM1) FOR READ.
                  411 ;               F0=0 SAYS TO CPL CHARS (FOR 9 DIGIT DISPLAY)
                  412 ;
                  413 ;****************************************************************
                  414         ;
00EF 18           415 WRTDSP: INC     R0              ;INC RAM PTR (BAM PTR INCED BY
                  416                                 ;REDBM1)
00F0 343F         417         CALL    REDBM1          ;READ FROM 1ST BAM
00F2 530F         418         ANL     A,#CLRMSN       ;GET READY TO DO MOVP3 ON
                  419                                 ;BASIC OF CHAR-(DO TABLE LOOKUP)
                  420                                 ;OF PAGE 3
00F4 E3           421         MOVP3   A,@A            ;GET ENCODED 7 SEG CODE FOR 4 CHAR
00F5 B6F8         422         JF0     WTDSP1          ;JUMP IF FOR 4 DIGIT
00F7 37           423         CPL     A
00F8 A0           424 WTDSP1: MOV     @R0,A           ;PUT IN RAM
00F9 EEEF         425         DJNZ    R6,WRTDSP
00FB 83           426         RET                     ;RETURN WHEN DONE
                  427 ;****************************************************************
                  428 ;
                  429 ;       DISFRG--THIS S/R DISENABLES THE FOREGROUND WHILE USING PORT 2
                  430 ;               FOR OTHER THINGS (MAINLY THE BAM)
                  431 ;
                  432 ;****************************************************************
                  433         ;
00FC 65           434 DISFRG: STOP    TCNT
00FD B81A         435         MOV     R0,#R2FORG      ;FOR INDIR ACCESS OF R2'
00FF 74C4         436         CALL    DSENP6          ;MAKES P6 = F HEX, LEAVES IN A
0101 3F           437         MOVD    P7,A
0102 F0           438         MOV     A,@R0           ;GET READY TO DISENABLE FOREGROUND
0103 4380         439         ORL     A,#080H         ;BY MAKING MSB OF R2 = 1
```

BAD ORIGINAL

```
LOC  OBJ      LINE        SOURCE STATEMENT

0105 A0        440 DSFRG1: MOV   @R0,A
0106 55        441         STRT  T
0107 83        442         RET
               443 ;**************************************************************
               444 ;
               445 ;       WRITE—
               446 ;                THIS S/R WRITES DATA TO THE BAM USING R0 AS AN ADDRESS
               447 ;                & THE DATA IN THE LSN OF THE ACCUMULATOR. BITS 0-7
               448 ;                OF R0 SELECT ONE OF 256 WORDS IN BAM-BITS P24 & P25
               449 ;                OF PORT 2 DECODE AS FOLLOWS
               450 ;                TO SELECT BAM 1 OR 2
               451 ;                       00-   BAM 1 & BAM 2
               452 ;                       01-   BAM 2
               453 ;                       10-   BAM 1
               454 ;                       11-   NEITHER BAM
               455 ;                SEPARATE-ENTRY PTS. ALLOWS WRITING TO BAM1 OR BAM2
               456 ;                FIRST R0 IS WRITTEN TO P1 AND THEN CHECKED.
               457 ;                AFTER WRITING THE WRITTEN DATA IS CHECKED
               458 ;                THE RETURN IS MADE WITH R0 INCR'D & THE ACCUMULATOR RESTORED
               459 ;                R7 IS CLOBBERED BY THIS S/R. F0 IS RESTORED BY RETR. THE
               460 ;                FOREGROUND MUST BE DISABLED.  WRTBME-110 M/S, WRTBM1-102.5 M/S.
               461 ;
               462 ;*******************************************************************
               463         ;
0108 85        464 WRTBM1: CLR   F0
0109 2412      465         JMP   WRITB1
010B 85        466 WRTBME: CLR   F0
010C 95        467         CPL   F0              ;F0=1 SAYS WRITING TO BAM 2
010D 530F      468         ANL   A,#CLRMSN       ;NEED TO WRITE TO BOTH BAMS
010F AF        469         MOV   R7,A            ;SO NEED TO HAVE BOTH NIBBLES =
0110 47        470         SWAP  A               ;FOR BAM WRITE CHECK
0111 4F        471         ORL   A,R7
0112 9A40      472 WRITB1: ANL   P2, #SOLEN      ;(040H) KEEP SOLEN SIGNAL
0114 8A1F      473         ORL   P2, #PROG2F     ;ENABLE 8243-2 SO CAN WRITE WRITE DATA TO P4
0116 3C        474         MOVD  P4,A
0117 9A4F      475         ANL   P2,#SOLNPF      ;THIS PREPARES FOR WRITE TO BOTH BAMS
0119 B61D      476         JF0   WR0000          ;JUMP IF TO WRITE TO BOTH
011B 8A2F      477         ORL   P2,#EBAM1       ;THIS WRITE ONLY TO BAM1
011D AF        478 WR0000: MOV   R7,A            ;SAVE ACCUMULATOR IN R7
011E F0        479         MOV   A, R0           ;MOVE ADDRESS TO P1
011F 39        480         OUTL  P1,A
0120 09        481         IN    A,P1            ;CHECK THAT WROTE OUT TO P1 O.K.
0121 D0        482         XRL   A,R0
0122 C626      483         JZ    WR0001          ;JUMP IF O.K.
               484 FAULT:
0124 64EC      485         JMP   FAULT1          ;SOFT FAULT, JUST TURN OFF MOTORS AND SOL AND STOP
0126 65        486 WR0001: STOP  TCNT            ;TO AVOID FOREGROUND CHANGING P2
0127 00        487         NOP                   ;TO ALLOW TIMER OVERFLOW TO HAPPEN IF READY
0128 8A8F      488         ORL   P2,#WRITBA      ;(080H) DO WRITE
012A 9A7F      489         ANL   P2,#SWRIBA      ;(07FH) STOP WRITE & KEEP BAM MSB OF ADRESS
012C 55        490         STRT  T               ;RESTART TIMER
012D 08        491         INS   A,BUS           ;READ BAM SELECTED
012E DF        492         XRL   A,R7            ;COMPARE WRITTEN DATA WITH ACTUAL
012F B633      493         JF0   WRITB2          ;JUMP IF WROTE TO BOTH
0131 530F      494         ANL   A,#CLRMSN       ;ONLY INTERESTED IN LSN
```

```
LOC  OBJ        LINE        SOURCE STATEMENT

0133 C639       495 WRITB2: JZ      WRITB5          ;JUMP IF READ WHAT WROTE
0135 230F       496         MOV     A,#00FH         ;ELSE GET READY TO JUMP TO FAULT
0137 64E1       497         JMP     FAULT0
0139 FF         498 WRITB5: MOV     A,R7            ;RESTORE ACCUM
013A 18         499         INC     R0              ;INCREMENT THE CHAR POINTER
013B 93         500         RETR
                501 ;****************************************************************
                502 ;
                503 ;       READB--THIS S/R READS THE BAM USING THE ADR IN R1 & RETURNS THE
                504 ;               RESULT IN THE ACCUMULATOR
                505 ;               THERE ARE TWO ENTRY POINTS. REDBM1 WHICH READS FROM BAM 1
                506 ;               AND REDBMB WHICH READS FROM BOTH BAMS.  F0 IS RESTORED BY RETR
                507 ;               R1 IS INCREMENTED AFTER THE READ.  REDBMB AND REDBMS- 52.5 M/S.
                508 ;               REDBM1 - 57.5 M/S
                509 ;
                510 ;****************************************************************
                511         ;
013C 85         512 REDBMB: CLR     F0              ;F0=0 SAYS READING FROM BOTH BAMS
013D 2441       513         JMP     READB1
013F 85         514 REDBM1: CLR     F0              ;F0=1 SAYS READING FROM BAM 1
0140 95         515         CPL     F0
0141 9A40       516 READB1: ANL     P2,#SOLEN       ;(040H) KEEP SOLEN SIGNAL
0143 8A0F       517         ORL     P2,#EBAM12
0145 F9         518 RAB001: MOV     A,R1            ;PUT ADDR ON P1
0146 39         519         OUTL    P1,A
0147 09         520         IN      A,P1            ;CHECK THAT P1 = R1
0148 D9         521         XRL     A,R1
0149 9624       522         JNZ     FAULT           ;JUMP IF PROBLEM
014B 08         523         INS     A, BUS          ;READ NIBBLE
014C 19         524         INC     R1              ;INCR THE CHAR POINTER
014D B65B       525         JF0     RDBMS1          ;IF READING FROM BOTH BAMS DON'T JUMP
014F 93         526         RETR
                527         ;THIS SPECIAL COPY OF READB READS FROM THE WRITE ADDR
                528         ;
0150 9A40       529 REDBMS: ANL     P2,#SOLEN       ;KEEP SOLENOID
0152 8A2F       530         ORL     P2,#EBAM1       ;READS FROM BAM1 ONLY,
0154 F8         531         MOV     A,R0            ;GET ADDR FROM R0
0155 39         532         OUTL    P1,A            ;PUT OUT TO P1
0156 09         533         IN      A,P1            ;CHECK THAT P1 = R0
0157 D8         534         XRL     A,R0
0158 9624       535         JNZ     FAULT
015A 08         536         INS     A,BUS           ;READ DATA FROM BUS IN LSN
015B 530F       537 RDBMS1: ANL     A,#CLRMSN       ;CLEAR OUT MSN(FOR REDBM1)
015D 93         538         RETR
                539 ;****************************************************************
                540 ;
                541 ;       S/R CLRKYR -- CLEAR KEYBOARD REGISTER.  IT DOES THAT AND
                542 ;               FIXES UP KEYSTA (IN R3) AND CALL CHKTOT.  IT USES A,R0,
                543 ;               R1,R6,R7
                544 ;
                545 ;****************************************************************
                546         ;
015E B860       547 CLRKYR: MOV     R0,#BKBREG      ;GET R0 READY FOR WRTBM1
0160 BE0A       548         MOV     R6,#00AH        ;GET R6 READY FOR LOOP CTR
0162 27         549         CLR     A               ;WANT TO WRITE A TO REG
```

BAD ORIGINAL

```
LOC  OBJ         SOURCE STATEMENT

0163 3489   CLR000: CALL  WRTBM1     ;WRITE TO BAM
0165 EE63           DJNZ  R6,CLR000  ;DONE YET?
0167 B92C           MOV   R1,#REG09  ;FIX UP R3 (STATUS
                                     ;OF KEYBOARD TASK) IN MSN REG09
0169 F1             MOV   A,@R1
016A AB             MOV   R3,A       ;CCE10 FIXES UP LSN
016B BE05           MOV   R6,#005H   ;TEMP FOR FOLLOWING
016D F270           JB7   CLR001     ;IF BIT7=1, P.O. + NORM + /FCENTS
016F CE             DEC   R6         ;HENCE, ADD CHARS TO KBREG(1) VS (0)
0170 FE     CLR001: MOV   A,R6
0171 3489           CALL  WRTBM1     ;WRITE TO BKBREG(A)
0173 8419           JMP   CHKTOT     ;MOVES DP TO TDR, AR TO TAR, ADDS
                                     ;KB TO TAR, SUBS KP FROM TDR.
                                     ;ADDS DR AND AR AN COMPARES WITH
                                     ;TOTAL (COPY IN TEMP)
            ;****************************************************************
            ;
            ;       MOVREG--S/R WHICH MOVES A REGISTER FROM
            ;               BAM TO ANOTHER REGISTER IN BAM
            ;               R0 POINTS TO THE WRITE ADDR IN BAM
            ;               R1 POINTS TO THE READ ADDR IN BAM
            ;               TWO ENTRIES POINTS ARE PROVIDED, MOVRB1 MOVES TO BAM1
            ;               AND MOVRBB MOVES TO BAM1 AND 2
            ;               R6 AND R7 ARE BOTH CLOBBERED AS WELL AS THE ACCUM
            ;               MOVRGS TAKES 1.85 M/S (2.6 M/S INCLUDING FOREGROUND TIME)
            ;
            ;****************************************************************
                    ;
0175 85     MOVRB1: CLR   F0         ;F0 = 1 SAYS WRITE TO BAM1
0176 2486           JMP   MREB1
0178 85     MOVRBS: CLR   F0         ;THIS ENTRY POINT MOVES 3 REGS (29H) BYTES
0179 95             CPL   F0
017A BE29           MOV   R6,#029H
017C 2492           JMP   MREBL
017E 85     MOVRBB: CLR   F0
017F 95             CPL   F0         ;F0=1 SAYS WRITE TO BAM2
0180 F8     MREB1:  MOV   A,R0       ;1ST MAKE WRITTEN REGISTER DIRTY
0181 AE             MOV   R6,A       ;SAVE TO RESTORE R0 LATER
0182 4309           ORL   A,#009H    ;THE 10TH NIBBLE MSB IS DIRTY BIT
0184 A8             MOV   R0,A
0185 07             DEC   A          ;TO GET LSN OF A=8 (DIRTY BIT)
0186 B68C           JF0   MREB2      ;JMP IF TO WRITE TO BAM2
0188 3489           CALL  WRTBM1
018A 248E           JMP   MREB3
018C 348E   MREB2:  CALL  WRTBMB
018E FE     MREB3:  MOV   A,R6
018F A8             MOV   R0,A       ;GET BACK ORIGINAL R0
0190 BE09           MOV   R6,#009H   ;GET LOOP CTR READY
0192 343F   MREBL:  CALL  REDBM1     ;READ FROM BAM USING R1
0194 B69A           JF0   MREB4      ;JMP IF TO WRITE TO BAM1 & 2
0196 3489           CALL  WRTBM1
0198 249C           JMP   MREB5
019A 348E   MREB4:  CALL  WRTBMB
                                     ;R0 AND R1 INCR BY WRITB AND REDBM1
019C EE92   MREB5:  DJNZ  R6,MREBL   ;DONE YET?
```

BAD ORIGINAL

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V3.0                PAGE  12
FME 925X METER, 4-10-79

LOC  OBJ        LINE        SOURCE STATEMENT

019E 27         605         CLR     A              ;MUST CLEAR OUT DIRTY REGISTER BIT
019F 3400       606         CALL    WRTBMB         ;POTENTIAL KLUDGE HERE (BOTH)
01A1 F9         607         MOV     A,R1           ;YES, REESTABLISH R1 FOR POSSIBLE
01A2 53F0       608         ANL     A,#CLR1SH      ;REUSE BY CALLING ROUTINE
01A4 A9         609         MOV     R1,A
01A5 F8         610         MOV     A,R0           ;REESTABLISH R0 FOR POSSIBLE REUSE
01A6 53F0       611         ANL     A,#CLR1SH      ;BY CALLING ROUTINE
01A8 A8         612         MOV     R0,A
01A9 93         613         RETR
                614 ;*********************************************************************
                615 ;
                616 ;       TOPTI--A S/R WHICH TURNS OFF THE PRINTER TIMER INTERRUPT BIT
                617 ;              AND STARTS THE PRINTER TIMER.
                618 ;
                619 ;*********************************************************************
                620         ;
01AA 2304       621 TOPTI:  MOV     A,#STINTV
01AC B836       622 TOPTIC: MOV     R0,#PRCTR
01AE 65         623         STOP    TCNT
01AF A0         624         MOV     @R0,A          ;RESTART PRINTER CTR
01B0 FA         625         MOV     A,R2
01B1 539F       626         ANL     A,#NPETTI      ;TURN OFF TIMER BIT
01B3 AA         627         MOV     R2,A
01B4 55         628         STRT    T
01B5 83         629         RET
                630 ;*********************************************************************
                631 ;
                632 ;       KBTLG--ROUTINE WHICH CHECKS IF THE KEYBOARD VALUE WILL CAUSE DR TO:
                633 ;              1--BE NEGATIVE IF IN MODE WHICH ALLOWS TO USE ALL MONEY
                634 ;                    IN DR
                635 ;              2--IF IN UPS MODE, NO CHECK DONE
                636 ;
                637 ;*********************************************************************
                638         ;
01B6 B878       639 KBTLG:  MOV     R0,#BTDRP8     ;THIS POINTS TO TDR(8) FOR LATER
01B8 B917       640         MOV     R1,#METYPE     ;GET COPY OF METYPE
01BA F1         641         MOV     A,@R1
01BB B2C1       642         JB5     KBTLG4         ;JUMP IF UPS MODE
01BD 3450       643         CALL    REDBMS
01BF 96C3       644         JNZ     KBTLG3         ;JUMP IF BORROWED FROM
01C1 4450       645 KBTLG4: JMP     KEYDF          ;KB O.K
01C3 B830       646 KBTLG3: MOV     R0,#030H       ;POINT TO 1ST CHAR OF 4 CHAR DISPLAY
01C5 10         647         INC     @R0            ;ADD OUT OF MONEY LIGHT
01C6 345E       648         CALL    CLRKYR         ;CLEAR THE KEYBOARD REG
01C8 34F8       649         CALL    TOFFSS         ;TURN OFF SOLENOID
01CA B935       650         MOV     R1,#PRTSTA     ;CLEAR THE PRINTER STATUS
01CC A1         651         MOV     @R1,A          ;TOFFSS RETURN WITH A = 0
01CD 230E       652         MOV     A,#00EH        ;FOR KEYBOARD-FULL
01CF 4455       653         JMP     FINNO8
                654 ;*********************************************************************
                655 ;
                656 ;       SND--THIS S/R SENDS THE CURRENT COMMAND TO THE APPROPRIATE
                657 ;              STEPPER
                658 ;
                659 ;*********************************************************************
```

BAD ORIGINAL

```
LOC  OBJ        LINE        SOURCE STATEMENT

                660         ;
01D1 34AA       661 SND:    CALL    TOPTI       ;THIS CLEARS TIMER TASK BIT
                662                             ;AND RESTARTS PRINTER TIMER
01D3 FC         663         MOV     A,R4        ;GET CURR COMMAND TO R7 (TEMP)
01D4 47         664         SWAP    A
01D5 AF         665         MOV     R7,A
01D6 9A4F       666         ANL     P2,#SOLEN   ;SAVE SOLENOID
01D8 8A20       667         ORL     P2,#PROG1   ;GET READY TO SEND OUT STEPPER COMMANDS
01DA FB         668         MOV     A,R3        ;SEE WHICH STEPPER TO SND TO
01DB 32E5       669         JB1     SND2
01DD 12E2       670         JB0     SND1
01DF FF         671 SND0:   MOV     A,R7        ;GET STEPPER COMMAND
01E0 3C         672         MOVD    P4,A        ;SEND TO IT'S 8279 PORT
01E1 83         673         RET
01E2 FF         674 SND1:   MOV     A,R7
01E3 3D         675         MOVD    P5,A
01E4 83         676         RET
01E5 12EA       677 SND2:   JB0     SND3
01E7 FF         678         MOV     A,R7
01E8 3E         679         MOVD    P6,A
01E9 83         680         RET
01EA FF         681 SND3:   MOV     A,R7
01EB 3F         682         MOVD    P7,A
01EC 83         683         RET
                684 ;*************************************************************
                685 ;
                686 ;
                687 ;       NORINT--THIS IS THE NORMALL INTERRUPT S/R. IT TURNS OFF
                688 ;               THE DISPLAY AND SOLENOID AND STEPPERS AND FIXES
                689 ;               UP THE STATE TO POWER DOWN GRACEFULLY;
                690 ;
                691 ;*************************************************************
                692         ;
                693         ;COME HERE BECAUSE PWRLSS = 1 (/INT = 0)
                694         ;ALREADY DONE SEL RB1 AND MOV R6,A (SAVED THE ACCUM)
                695         ;
01ED 15         696 NORINT: DIS     I           ;TURN OFF INTERRUPT
01EE 14FC       697         CALL    DISFRG      ; DISABLE THE DISPLAYS
                698                             ;(JUST KEEPS UP THE DECR OF PRCTR
                699                             ;AND THE FIRING OF
                700                             ;THE ONESHOTS FOR THE FAULT FFS)
01F0 34F8       701         CALL    TOFFSS      ;TURN OFF STEPPERS AND SOLENOID
01F2 B882       702         MOV     R0,#BAKSTA  ;WANT TO MAKE BAKSTA(7)=1
01F4 B080       703         MOV     @R0,#PFTSK
01F6 FE         704         MOV     A,R6        ;GET BACK ACCUM AND RETURN
01F7 93         705         RETR
01F8 8A40       706 TOFFSS: ORL     P2,#SOLEN   ;TURN OFF SOLENOID
01FA 9A40       707 TOFFST: ANL     P2,#SOLEN   ;KEEP SOLENOID IF USED BY PRINT
01FC 8A20       708         ORL     P2,#PROG1   ;GET READY TO TURN OFF STEPPERS
01FE 27         709         CLR     A           ;0 TO THE STEPPERS TURNS OFF THEIR POWER
01FF 3C         710         MOVD    P4,A
0200 3D         711         MOVD    P5,A
0201 3E         712         MOVD    P6,A
0202 3F         713         MOVD    P7,A
0203 83         714         RET
```

BAD ORIGINAL

```
 LOC  OBJ         LINE        SOURCE STATEMENT

                          ;*************************************************************
                          ;
                          ;
                          ;       KEYBD--THIS CO-ROUTINE IS JUMPED TO FROM MAIN IF BAYSTA(3)
                          ;               =1   THE FLOW THRU THIS ROUTINE IS CONTROLLED BY
                          ;               THE LSN OF BAYSTA   IT IS CALLED THE CURRENT
                          ;               CHARACTER AND IS = TO THE CURRENT KEYBOARD SWITCH
                          ;               BEING PRESSED .  SOMETIMES IT CONTAINS INFO ABOUT
                          ;               SWITCHES THAT HAVE BEEN PRESSED.  IF NOTHING
                          ;               DEPRESSED, CC=10 (0AH), CC=12 SAYS DISPLAYING ALL SEGS,
                          ;               CC=14 SAYS KEYBOARD REG FULL
                          ;
                          ;*************************************************************
                                  ;
                                  ;R3 IS INITED WITH KEYSTA
                                  ;
 0204 14FC-               KEYBD:  CALL    DISFRG          ;DISABLE FOREGROUND DURING KEYBOARD
 0206 B92A                        MOV     R1,#REG07       ;INIT R1 TO REG REG07
 0208 F1                          MOV     A,@R1
 0209 AE                          MOV     R6,A            ;PLACE IN R6
 020A B92C                        MOV     R1,#REG09       ;INI FOR LATER
 020C 85                          CLR     F0              ;FOR LATER USE
 020D FB                          MOV     A,R3            ;GET KEYSTA
 020E 7212                        JB3     KEY000
 0210 441B                        JMP     DISCHA          ;IF 7 OR LESS NO SPEC STATUS
 0212 5216                KEY000: JB2     KEY001
 0214 441B                        JMP     DISCHA          ;ITS 8 TO 11, NOT SPEC STATUS
 0216 321A                KEY001: JB1     KEY002
 0218 44EF                        JMP     DISPAL          ;CC=12, SEE IF STILL 7+8+9
 021A 95                  KEY002: CPL     F0              ;CC=14 SAVE IN F0
                                  ;
                                  ;DISCHA IS DISCOVER CHARACTER   IT LOOKS FOR A CHAR IN THIS ORDER
                                  ;7,8,9,4,5,6,1,2,3,CLRBATCH,0,CLR KEYBD.  THIS IS DONE TO TRY TO
                                  ;AVOID THE FAT FINGER PROBLEM WITH A RIGHT-HANDED PERSON   R1 HAS
                                  ;ADDR OF REG09
                                  ;
 021B F1                  DISCHA: MOV     A,@R1           ;GET REG09
 021C 2E                          XCH     A,R6            ;R6 HAS REG07 IN IT
 021D BF07                        MOV     R7,#007H        ;R7 CARRIES BCD CHAR BEING CHECKED
 021F F276                        JB7     CHK789          ;IF CHAR 7 DO WE HAVE 8 AND 9 ALSO?
 0221 2E                          XCH     A,R6            ;ELSE LOOK TO SEE IF EIGHT OR NINE
 0222 1F                          INC     R7              ;7 CONTAINS THE BCD OF THE CHAR BEING
                                                          ;CHECKED
 0223 1267                        JB0     FINNO           ;JMP IF '8'
 0225 1F                          INC     R7
 0226 3267                        JB1     FINNO           ;JMP IF '9'
 0228 2E                          XCH     A,R6            ;LOOK AT 4,5,6,ETC
 0229 BF04                        MOV     R7,#04
 022B 9267                        JB4     FINNO
 022D 1F                          INC     R7
 022E B267                        JB5     FINNO
 0230 1F                          INC     R7
 0231 D267                        JB6     FINNO
 0233 BF01                        MOV     R7,#001H
 0235 3267                        JB1     FINNO
```

BAD ORIGINAL

```
LOC  OBJ         LINE       SOURCE STATEMENT

8237 1F                     INC   R7
8238 5267                   JB2   FINNO
823A 1F                     INC   R7
823B 7267                   JB3   FINNO
823D 2E                     XCH   A,R6          ;GET REG89 TO CHECK CLEAR BATCH
823E 7292                   JB3   FINBAT        ;CLEAR BATCH IS DEPRESSED
8240 2E                     XCH   A,R6          ;CHECK BIT '0'
8241 BF00                   MOV   R7,#000H
8243 1267                   JB0   FINNO
8245 2E                     XCH   A,R6          ;CHECK CLEAR KEYBOARD
8246 52E4                   JB2   FNCLK5
8248 B82D                   MOV   R0,#LSTC89    ;NO KEYBOARD SW. DOWN, SEE IF CHANGE TO
824A F0                     MOV   A,@R0         ;P.D. MODE
824B 92E4                   JB4   FNCLK5        ;JUMP IF YES
824D FB                     MOV   A,R3          ;NOTHING IS DEPRESSED
                                                ;IF CC=10 OR LESS, MAKE CC=10.
824E 0306                   ADD   A,#006H       ;OTHER WISE LEAVE THE SAME
8250 C7                     MOV   A,PSW
8251 D258                   JB6   KEYDF         ;JMP IF AUX CARRY = 1
8253 230A            CCE10: MOV   A,#00AH       ;MAKE CC=10
8255 B803           FINNO8: MOV   R0,#003H      ;INIT R0 FOR XCHD
8257 30                     XCHD  A,@R0
8258 FB              KEYDF: MOV   A,R3
8259 B834                   MOV   R0,#KEYSTR    ;REPLACE KEYSTR WITH R3
825B A0                     MOV   @R0,A
825C B835                   MOV   R0,#PRTSTR    ;IF DOING CONSECUTIVE ENVELOPES.
825E F0                     MOV   A,@R0         ;PRTSTR WILL NO BE ZERO
825F C663                   JZ    MAINJ2        ;JUMP IF NOT DOING CONSEC. ENVS
8261 E479                   JMP   MAINR2        ;ELSE JUST GO BACK TO MAIN
8263 23F2           MAINJ2: MOV   A,#NKYIOP     ;GET READY FOR MAINR
8265 E476                   JMP   MAINR
                            ;
                            ;FOUND A NUMBER KEY DEPRESSED, IT'S BCD CODE IS IN R7, IF
                            ;CC=0-9, IT STAYS THE SAME VALUE. IF CC=10 THEN CC= NEWVALUE
                            ;CC=12 NOT POSSIBLE HERE, AND CC=14 AVOIDS THIS CODE.
8267 B663            FINNO: JF0   MAINJ2        ;IF CC=14 NOTHING TO DO
8269 FB                     MOV   A,R3
826A 530F                   ANL   A,#CLRMSN
826C 03F6                   ADD   A,#0F6H       ;0AH + 0F6H = 0
826E 9658                   JNZ   KEYDF         ;JUMP IF NOT = TO 0AH
8270 FF                     MOV   A,R7          ;CC=10, SO MAKE CC=NEW CHAR
8271 B803           FINNO1: MOV   R0,#003H      ;R7 HAS IT, GET READY FOR XCHD
8273 30                     XCHD  A,@R0
8274 6431                   JMP   AKBREG
                            ;
                            ;CHK789 CHECK IF HAVE 7*8*9, ALREADY HAVE FOUND '7', R7 HAS 07H IN
                            ;IT FOR FINNO AND R6 HAS REG 89
                            ;
8276 2E             CHK789: XCH   A,R6          ;
8277 37                     CPL   A             ;FOR EASE OF TESTING
8278 1267                   JB0   FINNO
827A 3267                   JB1   FINNO
827C 27                     CLR   A             ;DISPLAY ALL SEGMENTS, INI ACCUM
827D BF09                   MOV   R7,#009H      ;TO DISPLAY ALL CHARS IN 9 CHAR DISP
827F B920                   MOV   R1,#020H      ;R7 IS LOOP CTR, R1 IS POINTER TO
```

```
LOC  OBJ        LINE        SOURCE STATEMENT

                825                                 ;DECODED 7SEG CODES FOR 9 CHAR
0281 A1         826 DISALP: MOV     @R1,A           ;PUT OUT CHAR SEGMENTS
0282 19         827         INC     R1
0283 EF81       828         DJNZ    R7,DISALP       ;DONE?
                829                                 ;SEE IF A = 0 OR -1   IF -1
                830                                 ;READY TO RETURN, ELSE, MUST DO
                831                                 ;4CHAR DISPLAY
0285 F28E       832         JB7     DSALP1
0287 B930       833         MOV     R1,#030H        ;READY TO DISPLAY ALL SEGS FOR 4CHAR
0289 BF04       834         MOV     R7,#004H        ;STARTS AT 030H, HAS 4 CHARS,
028B 07         835         DEC     A               ;NEEDS -1 TO DISPLAY ALL
028C 4481       836         JMP     DISALP
028E 230C       837 DSALP1: MOV     A,#00CH         ;MAKE A = 12 FOR CC=12
                838                                 ;WILL CLEAR KB AT END OF 7*8*9
0290 4455       839         JMP     FINN08
                840         ;
                841         ;FINBAT--CAME HERE BECAUSE FOUND CLR BATCH DEPRESSED.
                842         ;IF P.O. MODE WE ARE READY TO ADD OR SUBR FROM DREG,
                843         ;IF NOT P.O. LOOKING FOR MOMENTARY DEPRESSION OF CLEAR KB
                844         ;TO INDICATE IT'S REALLY CLEAR BATCH   IN EITHER CASE, F0 SAYS CLR KB
                845         ;WAS DEPRESSED WHILE CLR BATCH DOWN   F0 = 1 IN P.O. SAYS SUBTRACT,
                846         ;F0 IN /P.O. SAYS DO CLEAR BATCH   NO F0 SAYS DO NOTHING
                847         ;
0292 85         848 FINBAT: CLR     F0              ;USE F0 AS FLAG TO TELL WHETHER
                849                                 ;TO ADD OR SUB K/B FROM DREG
                850                                 ;R1 IS INITED TO REG39 WHICH WE NEED
                851                                 ;TO READ TO SEE CHANGES IN CLR BATCH
                852                                 ;AND CLR KEYBOARD
0293 74D5       853         CALL    ENAFRG          ;NEED TO SEE CHANGES IN KEYBOARD
0295 F1         854 POBATL: MOV     A,@R1           ;GET NEW COPY OF SWITCHES
0296 B69D       855         JF0     POBAT1          ;IF F0 ALREADY SET DON'T CHECK
0298 529C       856         JB2     POB000          ;IS CLRKEYBD = 1
029A 449D       857         JMP     POBAT1          ;NO
029C 95         858 POB000: CPL     F0              ;YES, MAKE F0 = 1
029D 7295       859 POBAT1: JB3     POBATL          ;IS CLR BAT RELEASED?
029F 14FC       860         CALL    DISFRG          ;YES, DISABLE FOREGROUND AGAIN
02A1 FB         861         MOV     A,R3            ;SEE IF P.O. MODE
02A2 37         862         CPL     A               ;FOR EASE OF JMP
02A3 92D5       863         JB4     CLRBAT          ;JMP IF NOT
02A5 B87A       864         MOV     R0,#BTDREG      ;THIS IS P.O. ADD/SUB
02A7 74A0       865         CALL    ARKBNC          ;ADD/SUB TDREG AND KBD TO TDREG
02A9 C8         866         DEC     R0              ;R0 POINTS TO TDREG(9), WANT TDREG(8)
02AA 3458       867         CALL    RETBNS          ;IF NOT ZERO, TDREG IS GTR THAN
02AC C6B0       868         JZ      POADD1          ;$100,000. OR LESS THAN $0.0
02AE 24C3       869         JMP     KBTLG3          ;THIS LIGHTS OUT OF MONEY AND CLRS KB
02B0 B898       870 POADD1: MOV     R0,#BTEMP       ;HAS COPY OF TOTAL IN IT
02B2 74A0       871         CALL    ARKBNC          ;ADD/SUB KEYBOARD TO THAT
02B4 B998       872         MOV     R1,#BTEMP
02B6 B898       873         MOV     R0,#BTEMP       ;GOING TO GET BTEMP(BAK2) = TOTAL
02B8 3478       874         CALL    MOVRBS          ;+/- KEYBOARD
02BA 85         875         CLR     F0              ;NOW WANT TO MAKE BTAREG(BAK1) =
02BB B880       876         MOV     R0,#BTAREG      ;BTAREG + (BTDREG +/- KEYBOARD)
02BD B978       877         MOV     R1,#BTDREG
02BF 74A2       878         CALL    ARRNC
02C1 B880       879         MOV     R1,#BTAREG
```

BAD ORIGINAL

```
LOC  OBJ        LINE     SOURCE STATEMENT

02C3 B896                MOV   R0,#BTEMP      ;NOW MOVE RESULTS TO BTEMP1
02C5 3475                CALL  MOVRB1
02C7 B990                MOV   R1,#BTEMP      ;CHECK THAT BTEMP 1 AND 2 =
02C9 7480                CALL  COMPPX         ;CHECK THAT THE SAME AFTER MATH
02CB D433                CALL  MVTDR          ;IF O.K., MOVE TDR TO DR
02CD B990                MOV   R1,#BTEMP      ;MOVE BTEMP TO BTOTAL
02CF B826                MOV   R0,#BTOTAL     ;CHECKS OF THESE MOVES IS DONE BY
02D1 347E                CALL  MOVRB2         ;CHKTOT IN CLRKYR
02D3 44E4                JMP   FNCLK5         ;JUMP TO CLEAR KEYBOARD REG
                         ;
                         ;THIS IS CODE TO CLEAR BATCH REGISTERS (IF F0=1)
                         ;
02D5 95           CLRBAT: CPL  F0             ;FOR EASE OF TEST
02D6 B663                JF0   MAINJ2         ;JUMP IF F0 WAS 0
02D8 345E         CRBAT1: CALL CLRKYR         ;CLEARS REGS AND MOVES REGS TO TEMPS
02DA B838                MOV   R0,#BBATL
02DC B960                MOV   R1,#BKBREG     ;TO CLEAR BATCH REGS, WILL MOVE
02DE 3475                CALL  MOVRB1         ;CLEARED KB REG TO BATCH TOTAL
02E0 B84A                MOV   R0,#BBACT      ;AND COUNT
02E2 3475                CALL  MOVRB1         ;ONLY 1 COPY OF EACH
02E4 345E         FNCLK5: CALL CLRKYR         ;MAY BE REDUNDANT
02E6 74CC                CALL  REGREG         ;MAY ALSO BE REDUNDANT
02E8 FB                  MOV   A,R3           ;SEE IF P.O. MODE = 1
02E9 9253                JB4   CCE10          ;NO
02EB 94EE                CALL  DISKBF         ;DISPLAY THE KEYBOARD REGISTER THEN
02ED 4453                JMP   CCE10          ;END
                         ;
                         ;DISPAL-- DISPLAYING ALL SEGS. SEE IF STILL 7 + 8 + 9
                         ;R6 HAS #REG07, R1 HAS PTR TO REG09
                         ;
02EF F1           DISPAL: MOV  A,@R1          ;GET REG09
02F0 1258                JB0   KEYDF          ;STILL '8' DOWN
02F2 3258                JB1   KEYDF          ;STILL '9' DOWN
02F4 2E                  XCH   A,R6           ;GET REG07
02F5 F258                JB7   KEYDF          ;STILL '7' DOWN
02F7 44E4                JMP   FNCLK5         ;TO DISPLAY REGISTERS
                         ;
                         ;THIS IS IN THE 4TH PAGE OF PROGRAM MEMORY, WHICH STARTS WITH THE
                         ;ENCODE FROM BCD TO 7SEGMENT DISPLAY
                         ;
0300                     ORG   300H
0300 FC           CHAREN: DB   0FCH
0301 60                  DB    060H
0302 DA                  DB    0DAH
0303 F2                  DB    0F2H
0304 66                  DB    066H
0305 B6                  DB    0B6H
0306 BE                  DB    0BEH
0307 E0                  DB    0E0H
0308 FE                  DB    0FEH
0309 E6                  DB    0E6H
                         ;
                         ;*************************************************************
                         ;
                         ;      COMPBA--COMPARES EITHER A REGISTER OR ONE NIBBLE BETWEEN THE
```

BAD ORIGINAL

```
LOC  OBJ        LINE        SOURCE STATEMENT

                      ;            TWO BAYS, OR AS MANY NIBBLES AS WANT IF USE CMPLOT
                      ;            IF ENTER AT COMPEM, F1 IS SET, TELLS WHICH TYPE OF FAULT
                      ;            IF GET NOT COMPARE   F1=1 SAYS DOING MATH COMPARE, = 0
                      ;            SAYS DOING MOVE COMPARE
                      ;
                      ;************************************************************************
                      ;
030A A5               COMPEM: CLR  F1
030B B5                       CPL  F1            ;SIGNAL TO FAULT STUFF IN THIS ROUTINE
030C 6416                     JMP  CMPRA0
030E BE01             COMPEN: MOV  R6,#001H      ;GET READY TO DO NIBBLE
0310 A5               CMPLOT: CLR  F1
0311 85                       CLR  F0
0312 95                       CPL  F0            ;SAYS TO NOT CHANGE R1 AT END
0313 6419                     JMP  CMPRA1
0315 A5               COMPRA: CLR  F1
0316 BE0A             CMPRA0: MOV  R6,#00AH      ;GET READY TO DO REGISTER
0318 85                       CLR  F0            ;SAYS TO MAKE R1 POINT TO
                                                 ;BEGINNING OF REGISTER COMPARED
0319 343C             CMPRA1: CALL REDBMB
031B AD                       MOV  R5,A          ;SAVE FIRST RESULT IN R5
031C 47                       SWAP A             ;RESULTS OF READ IN BOTH NIBBLES
                                                 ;SHOULD BE SAME, SWAP AND XRL
031D DD                       XRL  A,R5          ;COMPARE
031E C627                     JZ   CMPRA3        ;ERROR IF NOT JUMP
0320 2306                     MOV  A,#006H
0322 7625                     JF1  CMPRA4
0324 17                       INC  A             ;INCR A IF MOVE COMPARE ERROR
0325 64E1             CMPRA4: JMP  FAULT0
0327 EE19             CMPRA3: DJNZ R6,CMPRA1
0329 B62F                     JF0  CMPRA2
032B F9                       MOV  A,R1          ;F0=0, CHANGE R1 TO REG(0)
032C 53F0                     ANL  A,#CLRLSN
032E A9                       MOV  R1,A
032F FD               CMPRA2: MOV  A,R5          ;GET LAST VALUE INTO A
0330 93                       RETR
                      ;************************************************************************
                      ;
                      ;       AKBREG--ROUTINE WHICH ADDS NEW CURR CHAR TO KEYBOARD REGISTER
                      ;            AND THEN COMPUTES IF THE REGISTER IS FULL AND THEN
                      ;            DISPLAYS THE NEW REGISTER DEPENDING UPON P.O. MODE
                      ;            IF NOT P.O. ALSO GETS NEW COPIES OF TDREG AND TAREG
                      ;            FROM AREG AND DREG USING CHKTOT THEN ADDS KB REG TO TAREG
                      ;            AND SUBS KB REG FROM TDREG AND CHECKS THAT TAREG + TDREG
                      ;            = A TEMP COPY OF TOTAL BROUGHT OVER BY CHKTOT, THEN DISPLAYS
                      ;            NEW KEYBOARD REG AND CHECKS THAT THE KEYBOARD REG VALUE ISN'T
                      ;            TOO LARGE
                      ;            THE MOST SIGNIFICANT BIT OF THE KEYBOARD REGISTER (AH)
                      ;            CONTAINS THE NUMBER OF CHARACTERS IN THE KEYBOARD
                      ;            REG + 5.  THIS IS FOR EASE OF TESTING.  ALSO MUST BE
                      ;            DECIDED WHERE TO ADD THE NEW CHARACTER (POSITION 1 OR 0).
                      ;            THIS IS DECIDED ON THE BASIS OF R3(7), WHICH IS P.O.
                      ;            MODE  + FCENTS*NORM
                      ;
                      ;************************************************************************
```

```
LOC  OBJ        LINE        SOURCE STATEMENT

                            ;
                            ;USES R0,R1,R6,R7,R4,R5
                            ;
0331 B82A            AKREG:  MOV     R0,#CURCHS       ;INI R0 WITH ADDR OF CURR KB CHAR
0333 345F                    CALL    REDBM1           ;GET IT
0335 17                      INC     A                ;INCREMENT IT
0336 AC                      MOV     R4,A             ;SAVE IT FOR LATER
0337 348B                    CALL    WRTBM1           ;WRITE IT TO BAM
0339 BE09                    MOV     R6,#009H         ;USE R6 AS LOOP COUNTER FOR LEFT
033B FE              AKREG1: MOV     A,R6
033C 4360                    ORL     A,#BKBREG        ;SHIFT OF KEYBOARD REGISTER
033E A8                      MOV     R0,A             ;INI R0 AND R1 FOR WRITE AND READB
033F A9                      MOV     R1,A             ;WRITE ADDR 1 GTR THAN READ ADDR
0340 C9                      DEC     R1
0341 343F                    CALL    REDBM1           ;GET CHAR
0343 348B                    CALL    WRTBM1           ;WRITE IT BACK
0345 EE3B                    DJNZ    R6,AKREG1        ;DONE YET?
0347 C8                      DEC     R0               ;TO OVERCOME AUTO INCR IN WRTBM1
0348 FB                      MOV     A,R3             ;IF P.O. + NORM + /FCENTS ADD TO
0349 F24C                    JB7     AKREG2           ;KBDREG(1) OTHERWISE KBDREG(0)
034B C8                      DEC     R0               ;THE NEW CHAR GOES INTO (0)
034C 348B            AKREG2: CALL    WRTBM1           ;WRITE OUT NEW CHAR
                            ;
                            ;SEE IF KEYBD REG IS FULL
                            ;
034E FC                      MOV     A,R4             ;IF LESS THAN 3 (5+X CHARS) NOT FULL
034F 7278                    JB3     AKREG7           ;JUMP IF MORE THAN 3 CHARS
0351 74CC            AKREG4: CALL    REGREQ           ;POSSIBLY REDUNDANT
                                                      ;ALSO COME HERE FROM END OF PRINT
0353 FB                      MOV     A,R3             ;SEE IF P.O. MODE = 1
0354 9276                    JB4     AKREG5           ;JUMP IF YES
0356 94D8                    CALL    CHKTOT           ;GET NEW COPIES OF DR AND AR
0358 B888                    MOV     R0,#BTAREG       ;F0 IS CLEARED BY CHKTOT
035A 74A8                    CALL    ARKBNC           ;ADD KB TO TAR
035C 95                      CPL     F0               ;TO DO SUBTRACT
035D B876                    MOV     R0,#BTDREG
035F 74AA                    CALL    ARKBNC           ;SUBTRACT KB FROM TDR
0361 B890                    MOV     R0,#BTEMP        ;NOW MUST CHECK TDREG + TAREG
0363 B988                    MOV     R1,#BTAREG       ;= COPY OF TOTAL IN BTEMP(BAM2)
0365 3475                    CALL    MOVR21           ;MOVE TAREG TO TEMP(BAM1)
0367 B890                    MOV     R0,#BTEMP
0369 B978                    MOV     R1,#BTDREG
036B 85                      CLR     F0               ;WANT TO ADD TDREG AND TAREG (IN TEMP)
036C 74A2                    CALL    ARRNC            ;DO ADD
036E B990                    MOV     R1,#BTEMP        ;DO COMPARISON
0370 740A                    CALL    COMPBM           ;FOR MATH
0372 948E                    CALL    DISKBR           ;SO DISPLAY NEW KEYBD REG ON 4CHAR
0374 248E                    JMP     KBTLG            ;JUMP TO SEE IF KEYBOARD TOO LARGE
0376 4459            AKREG5: JMP     KEYDF
                            ;
                            ;SEE IF KEYBD REG FULL, IF P.O. MODE, NEED TO SEE IF UP TO
                            ;MAX P.O VALUE, ENCODED IN LSN OF METYPE
                            ;  IF NOT P.O., IF FCENTS=1 OR IF 3 STEPPER METER
                            ;THEN WE ARE NOW FULL OTHERWISE IF 99.99 (BIT0=1), IT'S FULL
                            ;
```

BAD ORIGINAL

```
LOC  OBJ      LINE        SOURCE STATEMENT

0378 B817     1045 AKREG7: MOV   R0,#METYPE    ;INIT R0 TO GET MAX P.O VALUE AND
037A FB       1046         MOV   A,R3
037B 928C     1047         JB4   AKREGA        ;JUMP IF P.O. MODE
037D D285     1048         JB6   AKREG9        ;JUMP IF FCENTS
037F FC       1049         MOV   A,R4
0380 1285     1050         JB0   AKREG9        ;OR THIS TO MAKE KEYBD REG FULL
0382 F0       1051         MOV   A,@R0         ;LOOK TO SEE IF 3 OR 4 STEPPERS
0383 F251     1052         JB7   AKREG4        ;JUMP IF 4
0385 B803     1053 AKREG9: MOV   R0,#003H      ;GET READY TO MAKE LSN OF R3
0387 230E     1054         MOV   A,#00EH       ;= 14D USING XCHD
0389 30       1055         XCHD  A,@R0
038A 6451     1056         JMP   AKREG4
038C F0       1057 AKREGA: MOV   A,@R0         ;GET MAX P.O. VALUE
038D DC       1058         XRL   A,R4          ;DO XOR WITH CURR KBCC
038E 530F     1059         ANL   A,#CLRMSN
0390 C685     1060         JZ    AKREG9        ;IF FULL, JUMP
0392 6451     1061         JMP   AKREG4        ;ELSE JUMP HERE TO END
              1062 ;************************************************************
              1063 ;
              1064 ;       DONE -- S/R WHICH DOES THE END COMPARISON OF THE FOREGROUND
              1065 ;               REGISTERS R4,R5,R6 TO DO THE DEBOUNCING OF SWITCHES.
              1066 ;               IT IS CALLED AFTER THE REGISTERS HAVE BEEN SET
              1067 ;               UP BY FDBCK AND THEY ARE REALIGNED.  THE LOGIC FOR ONE BIT
              1068 ;               IS AS FOLLOWS:  AN XOR IS DONE BETWEEN THE CURRENT
              1069 ;               VALUE (TO THE M/P) IN R4 AND THE NEW VALUE (GOTTEN
              1070 ;               FROM T1).  THIS IS DONE BY FDBCK.  IN R5 IS THE VALUE
              1071 ;               OF THIS COMPARISON LAST TIME THE SWITCH WAS CHECKED.
              1072 ;               THE REST OF THIS IS DONE BY THIS S/R.  NOW THE LAST
              1073 ;               COMPARISON IN R5 IS COMPARED WITH THIS TIME COMPARISON
              1074 ;               IN R6.  SO IF THERE WAS A CHANGE NOTICED BOTH TIMES,
              1075 ;               THIS LAST COMPARISON (AN AND) WILL BE ATRUE.  THIS
              1076 ;               VALUE IS NOW SAVED IN R5 (FOR LATER USE BY TIMINT)
              1077 ;               IF IT IS NONZERO.  ALSO F1 IS SET TO INDICATE THIS.
              1078 ;               NOW R4 (THE CURRENT M/P VIEW OF THE SWITCH IS XORED
              1079 ;               IF THE VALUE IN R5 IS NON ZERO.  ANY OF THE SWITCHES
              1080 ;               WITH THEIR EQUIVALENT BIT IN R5 BEING TRUE WILL BE
              1081 ;               CHANGED, FROM 1 TO 0 OR V.V.  R6 THE NEW LAST TIME
              1082 ;               CHANGE VALUES ARE STORED USING @R0 AND IF F1 WAS
              1083 ;               SET, THE NEW CURRENT VALUES OF THE DEBOUNCED SWITCHES
              1084 ;               ARE STORED BACK, AFTER DECREMENTING R0.  IN EITHER
              1085 ;               CASE, THE NEW CURRENT VALUES OF THE DEBOUNCED SWITCHES
              1086 ;               ARE RETURNED IN THE ACCUM WITH R0 POINTING TO WHERE
              1087 ;               THEY SHOULD BE STORED IF NECESSARY TO MAKE A CHANGE
              1088 ;               TO THESE REGISTERS BY THE DONEX ROUTINES.
              1089 ;
              1090 ;************************************************************
              1091         ;
0394 FE       1092 DONE:   MOV   A,R6          ;GET NEW LAST TIME CHANGES
0395 A0       1093         MOV   @R0,A         ;STORE IT
0396 C8       1094         DEC   R0            ;NOW POINTS TO M/P CURRENT VALUE
0397 5D       1095         ANL   A,R5          ;DO AND BETWEEN THIS TIME CHANGES
              1096                               ;AND LAST TIME CHANGES
0398 A5       1097         CLR   F1
0399 C69C     1098         JZ    FNOCHG        ;JUMP IF AND = 0, NO CHANGES
039B B5       1099         CPL   F1            ;ELSE COMPLEMENT F1 TO 1
```

0019515


```
LOC  OBJ     LINE     SOURCE STATEMENT

039C AD           FNOCHG: MOV   R5,A          ;SAVE THE VALUE IN THE ACCUM IN R5
                                              ;FOR LATER USE BY TIMINT
039D DC                   XRL   A,R4          ;DO XOR
039E AA                   MOV   @R0,A         ;STORE NEW CURRENT VALUES
039F 83                   RET
                  ;*************************************************************
                  ;
                  ;       ARKBNC--S/R WHICH ADDS A REGISTER IN R0 TO THE KEYBOARD REGISTER
                  ;             IF F0 = 1. IT SUBTRACTS THE KEYBOARD REGISTER FROM THE R0
                  ;             REGISTER   ARRNC TAKES 2.2 M/S (3.0 M/S INCL FOREG)
                  ;
                  ;*************************************************************
                          ;
03A0 B960         ARKBNC: MOV   R1,#BKBREG    ;INIT R1 TO THE KEYBOARD REG
03A2 95           ARRNC:  CPL   F0            ;LET F0 = 1 MEAN ADD
03A3 97                   CLR   C             ;CLEAR CARRY BIT
03A4 B6A7                 JF0   AR2           ;SET IF SUBTRACTING
03A6 A7                   CPL   C
03A7 BE09         AR2:    MOV   R6,#009H      ;INIT LOOP COUNTER
03A9 3450         ARITLP: CALL  REDEMS        ;THIS READS USING THE R0 POINTER
03AB AD                   MOV   R5,A          ;SAVE TEMP IN R5
03AC 343F                 CALL  REDEM1        ;THIS READS USING THE R1 PTR
03AE B6BA                 JF0   ARADD         ;JUMP IF ADDING
03B0 A5                   CLR   F1            ;SUBTRACTING, SAVE CARRY
03B1 E6B4                 JNC   AR3           ;IF NO CARRY, SET F1
03B3 B5                   CPL   F1
03B4 37           AR3:    CPL   A             ;GET 9'S COMPLEMENT
03B5 030A                 ADD   A,#010D       ;ALWAYS SETS CARRY
03B7 76BA                 JF1   ARADD         ;IF CARRY WS SET, KEEP IT
03B9 A7                   CPL   C
03BA 4390         ARADD:  ORL   A,#090H       ;SO THAT DA WILL SET CARRY IF CARRY
                                              ;OUT OF NIBBLE
03BC 7D                   ADDC  A,R5          ;DO ADD
03BD 57                   DA    A             ;DO DECIMAL ADJUST
03BE 3403         ARTLP1: CALL  WRTEM1        ;WRITE OUT
03C0 EEA9                 DJNZ  R6,ARITLP     ;DONE
03C2 95                   CPL   F0            ;RESTORE F0
03C3 83                   RET
                  ;*************************************************************
                  ;
                  ;       DSENP6--THIS S/R MOVES F HEX TO PORT 6 OF THE DISPLAY 8279.
                  ;               IT IS ALSO USED TO SET UP PORT 2 FOR THAT 8279
                  ;
                  ;*************************************************************
                          ;
03C4 23FF         DSENP6: MOV   A,#MINONE
03C6 9A40         P6MOVD: ANL   P2,#SOLEN     ;THIS ENTRY POINT ALLOWS ANY
03C8 8A10                 ORL   P2,#PROG2     ;VALUE TO BE MOVD'D TO PORT 6
03CA 3E                   MOVD  P6,A
03CB 83                   RET
                  ;*************************************************************
                  ;
                  ;       REGREQ--THIS S/R SETS UP A TASK (IN ACCUM) IN BAKSTA
                  ;
                  ;*************************************************************
```

BAD ORIGINAL

```
LOC  OBJ      LINE    SOURCE STATEMENT

                      ;
03CC 2310     REGRED: MOV    A,#SWTSK       ;ENTRY FOR REGISTER REQUEST
03CE B902     BAKRED: MOV    R1,#BAKSTA     ;ENTRY FOR OTHER REQUESTS
03D0 65               STOP   TCNT
03D1 41               ORL    A,@R1          ;GET BAKSTA
03D2 A1               MOV    @R1,A          ;STORE BACK
03D3 55               STRT   T
03D4 83               RET
              ;*****************************************************************
              ;
              ;       ENAFRG--S/R TO REENABLE THE FOREGROUND AFTER USING DISFRG
              ;
              ;*****************************************************************
                      ;
03D5 65       ENAFRG: STOP   TCNT
03D6 B81A             MOV    R0,#R2FORG     ;FOR INDIR ACCESS OF R2'
03D8 F0               MOV    A,@R0          ;GET READY TO REENABLE THE FOREGROUND
03D9 537F             ANL    A,#07FH        ;GET RID OF MSB
03DB 2405             JMP    DSFRG1
              ;*****************************************************************
              ;
              ;       FAULTC--ROUTINE TO SEE IF HOME FAULT OR CLUTCH HALT
              ;
              ;*****************************************************************
                      ;
              FAULTC:                       ;SET UP TO R7' COMPLEMENTED
03DD 92EC             JB4    FAULT1         ;JUMP IF NOT HOME TO SOFT FAULT
03DF 2303             MOV    A,#003H        ;COME HERE FOR HOME FAULT
              FAULT0:                       ;IN CASE POWER LOSS
03E1 2A               XCH    A,R2           ;SEE IF POWER LOSS
03E2 F2EC             JB7    FAULT1         ;NO FAULT IF YES
03E4 2A               XCH    A,R2           ;ELSE GO ON AND DO FAULT STUFF
03E5 35               DIS    TCNTI          ;COME HERE IF FAULT
03E6 C6EC             JZ     FAULT1         ;IF ZERO FAULT, DON'T WRITE OUT
03E8 B82D             MOV    R0,#BFAULT
03EA 3400             CALL   WRTBMB
03EC 34F8     FAULT1: CALL   TOFFSS         ;TURN OFF SOLEN
03EE 64EE             JMP    $
              ;*****************************************************************
              ;
              ;       IOWRB--PART OF I/O ROUTINE, THIS WRITES TO BAM IFF TEST=1
              ;              ELSE ERROR.  IT WRITE TO BAM 1 IF BIT 4 OF DATA WORD
              ;              IS 0 , ELSE IT WRITE TO BOTH BAMS.  AFTER WORDS IT JUMPS
              ;              TO IOTSK5 WHICH INCREMENTS THE I/O POINTER
              ;
              ;*****************************************************************
                      ;
03F0 F1       IOWRB:  MOV    A,@R1          ;SET UP TO R7FORG
03F1 12F5             JB0    IOWRB1         ;JUMP IF TEST (LOADED IN IOTSK3) = 1
03F3 84CE             JMP    IOERR          ;ELSE JUMP TO ERROR
03F5 FD       IOWRB1: MOV    A,R5           ;ELSE GET DATA WORD
03F6 3400             CALL   WRTBM1         ;WRITE TO BAM1
03F8 92FD             JB4    IOWRB2         ;JUMP IF WRITE TO BAM1 ONLY
03FA C8               DEC    R0             ;ELSE DEC WRITE POINTER
03FB 3400             CALL   WRTBMB         ;AND WRITE TO BAM1 AND 2
```

```
LOC  OBJ        LINE        SOURCE STATEMENT

03F1 E4E0       1210 JDOFS2: JMP   IOTSIS       ;AND JUMP BACK TO INCREMENT POINTER
                1211         ;
0400            1212         ORG   400H
                1213         ;
                1214 ;***********************************************************
                1215
                1216 .       DONES--COME TO DONE 5 AFTER THE SEVERAL DISPLAYS OF 4 CHAR DISPLAY
                1217              WITH ALL 5 REG SWITCHES PARTIALLY DEBOUNCED. NEED TO DO THE
                1218 .            SAME FOR TAPE, ENV, REPEAT.  THEN CALL DONE WHICH DOES FINAL
                1219 .            DEBOUNCING AND THEN IF CHANGES TO REG SWE MODIFY BASSTA .
                1220 .
                1221 ;***********************************************************
                1222         ;
0400 BA04       1223 DONES:  MOV   R2,#004H     ;INIT R2
0402 9421       1224         CALL  FDBCK1       ;DOES INCR OF R2 AND MOVD TO P5
0404 9421       1225         CALL  FDBCK1       ;THEN FIRST  1ST TAPE, THEN ENVELOPE
0406 9421       1226         CALL  FDBCK1       ;THEN REPEAT
0408 7494       1227         CALL  DONE         ;DOES FINAL DEBOUNCING, IF F1=1
040A 7612       1228         JF1   DONES2       ;MAKE CHANGE
040C BA2F       1229 DONES1: MOV   R2,#02FH     ;DONE WITH 4CHAR SET UP FOR NINE CHAR
040E B82F       1230         MOV   R0,#REGB7    ;R0: FIRST OF KEYBD SWITCHES
0410 0414       1231         JMP   ONESHT       ;FIRE FAULT FF ONE SHOTS
0412 FD         1232 DONES2: MOV   A,P5         ;WANT TO SEE IF CHANGE IN
0413 5350       1233         ANL   A,#050H      ;TAPE OF ENVELOPE
0415 C61D       1234         JZ    DONES3       ;JUMP IF NONE
0417 2340       1235         MOV   A,#FTHEN     ;ADD FTHEN HOWEVER TO BASSTA
0419 7400       1236         CALL  BASREG
041B 040C       1237         JMP   DONES1
041D 7400       1238 DONES3: CALL  REGREG       ;EITHER REPEAT CHG OR REG CHG
041F 040C       1239         JMP   DONES1
                1240 ;***********************************************************
                1241 ,
                1242 ;       FDBCK--S/R FOREGROUND DEBOUNCE (KEYS)
                1243 ;            IT DOES THE BEGINNING DEBOUNCING BY DOING A COMPARISON
                1244 ;            BETWEEN R4 (REGSW) AND T1  IT EXPECTS THE 74150 (T1)
                1245 ;            ALREADY SET UP.  ALSO THE 74145 IF NEEDED  IT DOES
                1246 ;            AN EOR OF THE TWO BITS IN QUESTION (R4(0) AND T1).
                1247 ;            IF TRUE, THE LSB OF R6 IS SET  THEN R4 AND R6 ARE RR
                1248 ;            AND A RETURN DONE  THERE IS A ENTRY POINT CALLED
                1249 ;            FDBCK1 WHICH INCS R2 AND MOVDS IT TO P5  THERE IS ANOTHER
                1250 ;            ENTRY POINT WHICH MERELY DOES THE RR OF R4 AND R6
                1251 ;
                1252 ;***********************************************************
                1253         ;
0421 1A         1254 FDBCK1: INC   R2
0422 FA         1255         MOV   A,R2
0423 3D         1256         MOVD  P5,A         ;ENTRY POINT TO LOOK AT NEXT INPUT,
                1257                            ;OF 74150
0424 27         1258 FDBCK:  CLR   A
0425 4628       1259         JNT1  FDBK2
0427 17         1260         INC   A            ;MAKE ACCUM (0) = T1
0428 DC         1261 FDBK2:  XRL   A,R4         ;DO XOR
0429 37         1262         CPL   A            ;TO HELP IN PROGRAMMING
042A 122D       1263         JB0   FDBK3
042C 1E         1264         INC   R6           ;R6(0) = 1 IF SXOR OF R4(0)
```

BAD ORIGINAL

```
LOC  OBJ        LINE        SOURCE STATEMENT

                                                        ;AND T1 = 1
042D FE              FDBK1:  MOV     A,R6               ;DO RR OF R4 AND R6
042E 77                      RR      A
042F AE                      MOV     R6,A
0430 FC                      MOV     A,R4
0431 77                      RR      A
0432 AC                      MOV     R4,A
0433 83                      RET
                     ;*****************************************************************
                     ;
                     ;       REGDSP--A BACKGROUND TASK WHICH READS THE REGSW REG AND DISPLAYS
                     ;               THE 9 CHAR DISP.  IF P.O. MODE, AND BATCH COUNT OR BATCH TOTAL
                     ;               DO KEYBOARD
                     ;
                     ;*****************************************************************
                             ;
                             ;USES R5,R1,R0,R6,R7, AND R3 IS INITED TO KEYSTA
                             ;
0434 14FC            REGDSP: CALL    DISFRG
0436 B91F                    MOV     R1,#R7FORG         ;SEE IF ON METER BASE
0438 F1                      MOV     A,@R1
0439 723D                    JB3     RGD00A             ;JUMP IF YES, ELSE BLANK 4 CHAR DISP
043B 9484                    CALL    BLKKB              ;BLANK 4 DIGIT DISPLAY
043D B81F            RGD00A: MOV     R0,#01FH           ;GET R0 READY TO WORK WITH 9CHAR DISP
043F BE09                    MOV     R6,#009H           ;(ADDR-1) AND R6 READY AS CHAR CTR
0441 B92E                    MOV     R1,#REGSW          ;GET REGSW
0443 F1                      MOV     A,@R1
0444 531F                    ANL     A,#01FH            ;AND OUT BITS NOT REGISTER
0446 C675                    JZ      RGDSP4             ;IF ZERO (NO SWITCHES DOWN, JUMP
0448 AD                      MOV     R5,A               ;ELSE SAVE IN R5
0449 FB                      MOV     A,R3
044A 9279                    JB4     RGDSPP             ;JUMP IF P.O. MODE
044C FD                      MOV     A,R5               ;GET REGSW AND CONVERT TO
044D 2F00            RGD00E: MOV     R7,#0              ;BINARY FOR ADDR OF REG
044F 1255            RGD00D: JB0     RGDSP3             ;SEE IF THIS IS REG SELECTED
0451 77                      RR      A                  ;NO, SO RR A
0452 1F                      INC     R7                 ;AND INCR CTR
0453 844F                    JMP     RGD00D             ;AND SEE IF THIS IS REG SELECTED
0455 FF              RGDSP3: MOV     A,R7               ;GET BINARY #
0456 47                      SWAP    A                  ;PUT IN MSN FOR ADDR OF REG
0457 A9              RGDSP1: MOV     R1,A               ;GET READY FOR WRTDSP
0458 85                      CLR     F0                 ;FOR WRTDSP
0459 FB                      MOV     A,R3               ;IF /FCENTS, NEED TO UP R1 BY 1
045A D25D                    JB6     RGD003             ;JUMP IF FCENTS
045C 19                      INC     R1                 ;INCR R1
045D 14FF            RGD003: CALL    WRTDSP
                             ;
                             ;NOW HAVE TO DELETE LEADING ZEROS AND PUT IN DECIMAL POINT
                             ;R0 POINTS TO CHAR 28, R1HAS THE CHAR
                             ;COUNT, ADD 0FCH AFTER CLEARING MSN AAND THIS GIVES COUNT FOR
                             ;CHECKING FOR LEADING ZEROS
                             ;
045F F9                      MOV     A,R1
0460 530F                    ANL     A,#CLRMSN
0462 03FC                    ADD     A,#0FCH
```

BAD ORIGINAL

```
LOC  OBJ     LINE   SOURCE STATEMENT

0464 AF                     MOV   R7,A          ;MOVE LOOP CTR TO R7
0465 F0             RGD005: MOV   A,@R0         ;GET CHAR
0466 D303                   XRL   A,#ZERO9C     ;SEE IF = ZERO
0468 966F                   JNZ   RGD008        ;JUMP OUT OF LOOP IF NOT
046A 37                     CPL   A             ;ELSE COMPL A FOR A CHAR WHICH IS BLANK
046B A0                     MOV   @R0,A         ;STORE BACK IN SEG DECODE ARRAY
046C C8                     DEC   R0            ;TO LOOK AT NEXT CHAR
046D EF65                   DJNZ  R7,RGD005
                                                ;NOW USE THIS TO FIND WHERE
046F C8             RGD008: DEC   R0            ;TO ADD DECIMAL POINT
0470 EF6F                   DJNZ  R7,RGD008     ;WHEN DONE, R0 POINTS TO CHAR
0472 F0                     MOV   A,@R0         ;GET CHAR SEGMENTS
0473 07                     DEC   A             ;ADD DECIMAL PT DISPLAY
0474 A0                     MOV   @R0,A
0475 23EF           RGDSP4: MOV   A,#HSHTSK     ;GET READY FOR MAIN
0477 E476                   JMP   MAIN
0479 9484           RGDSPP: CALL  BLKKB         ;BLANK 4 CHAR DISPLAY
047B FD                     MOV   A,R5          ;P.O. MODE, IF REGSW = BATCH, DO
047C 5307                   ANL   A,#007H       ;DISPLAY OF KEYBOARD REG
047E 964D                   JNZ   RGD08E        ;JUMP IF NOT
0480 2360                   MOV   A,#BKBREG     ;DISPLAY KEYBOARD REG
0482 8457                   JMP   RGDSP1
0484 B930           BLKKB:  MOV   R1,#030H      ;GET READY TO CLEAR 4 CHAR DISPLAY
0486 27                     CLR   A
0487 BF04                   MOV   R7,#004H      ;GET LOOP CTR READY
0489 A1             RDG001: MOV   @R1,A         ;ZERO TO 4 CHAR DISPLAY = BLANK
048A 19                     INC   R1
048B EF89                   DJNZ  R7,RDG001
048D 83                     RET
                    ;*************************************************************
                    ;
                    ;        DISKBR--S/R WHICH DISPLAYS THE KEYBOARD REGISTER.  IT MUST WORRY
                    ;                ABOUT SUCH THINGS AS PLACEMENT OF DECIMAL POINTS,
                    ;                DISPLAYING A SERIAL, ETC.  IT CALLS
                    ;                WRTDSP (WRITE DISPLAY) WHICH IS A S/R WHICH READS A REGISTER
                    ;                (RAM) POINTED TO BY R1 AND WRITES TO RAM POINTED TO BY R0.
                    ;                F0=1 SAYS DON'T COMPL SEGMENTS AND R7 NEEDS THE # OF CHARS
                    ;                TO BE MOVED.  THE CHARACTERS PLACED IN RAM ARE FIRST CONVERTED
                    ;                TO 7 SEGMENT CODE FOR THE 9 CHAR DISPALY, HENCE THIS ROUTINE
                    ;                MUST COMPLEMENT THEM ADDING DECIMAL POINT WHERE APPROP.
                    ;
                    ;*************************************************************
                            ;
048E B960           DISKBR: MOV   R1,#BKBREG    ;PUT IN R1 THE ADDR OF THE KEYBD REG
0490 85                     CLR   F0
0491 95                     CPL   F0            ;MAKE F0=1 FOR LATER
0492 BE04                   MOV   R6,#004H      ;INI # OF CHAR CTR
0494 B82F                   MOV   R0,#02FH      ;R0 PTS TO BEGINNING OF 4 CHAR DISREG(-1)
0496 FB                     MOV   A,R3          ;LOOK AT WHETHER TO READ FROM KBREG(0)
                                                ;OR KBREG(1)
0497 D29A                   JB6   DSKBRB        ;IF JUMP, FROM (0)
0499 19                     INC   R1            ;IT'S /FCENTS, READ FROM (1)
049A 14EF           DSKBRB: CALL  WRTDSP        ;MOVE FROM KEYBD REG TO 4 CHAR DISPREG
                                                ;RETURNS WITH R0 PTING TO MSD OF
                                                ;4 DIGIT DISPLAY
```

BAD ORIGINAL

```
LOC  OBJ            SOURCE STATEMENT

049C FB                     MOV    A,R3          ;SEE IF FCENTS MODE
049D D2A1                   JB6    DSKBR1        ;JUMP IF YES
049F 85                     CLR    F0            ;FOR LATER TEST
04A0 C8                     DEC    R0            ;WANT TO PUT DEC TO 2ND MSDIGIT
04A1 10             DSKBR1: INC    @R0           ;ADD DEC PT TO CORRECT CHAR
04A2 95                     CPL    F0            ;FOR EASE OF TESTING
04A3 B6AD                   JF0    DSKBRF        ;IF FCENTS DON'T JUMP
04A5 FB                     MOV    A,R3          ;SEE IF .00 DISPLAY
04A6 B2A9                   JB5    DSKBRS        ;JUMP IF .00 DISPLAY
04A8 83                     RET
04A9 B830           DSKBRS: MOV    R0,#030H      ;IT'S FCENTS * .00
04AB B092                   MOV    @R0,#SERIPH   ;SO DISPLAY SERIPH
04AD 83             DSKBRF: RET
                    ;*****************************************************************
                    ;
                    ;      IOTSK3--PART OF THE IO ROUTINE, COMEHERE WHEN HAVE RECEIVED A
                    ;              DATA WORD FROM THE UPI.  STORE IT IN R5, CHECK THAT R4
                    ;              IS NOT ERROR COMMAND, THEN JUMP TO PROPER ROUTINE
                    ;
                    ;*****************************************************************
                           ;
04AE AD             IOTSK3: MOV    R5,A          ;STORE DATA IN R5
04AF 74C4                   CALL   DSENP6        ;TURN OFF ENAIO
04B1 FC                     MOV    A,R4          ;SEE IF ERROR COMMAND
04B2 F2D0                   JB7    ITSK1J        ;JUMP IF YES
04B4 52D2                   JB2    IOWRBJ        ;ELSE JUMP IF IOWRB
04B6 32BE                   JB1    IOTSK6        ;OR JUMP TO IOAKB
04B8 B937                   MOV    R1,#IOPTR     ;ELSE IT'S IORPTR
04BA FD                     MOV    A,R5          ;GET DATA WORD
04BB A1                     MOV    @R1,A         ;WRITE NEW POINTER
04BC E4CD                   JMP    IOEND2
04BE FD             IOTSK6: MOV    A,R5          ;IT'S IOAKB, SEE IF # OR CLEAR
04BF 03F6                   ADD    A,#0F6H
04C1 C6D6                   JZ     IOAKB1        ;IF ZERO, IT'S CLEAR KEYBOARD
04C3 07                     DEC    A
04C4 96C8                   JNZ    IOABK2        ;IF JUMP SHOULD BE #
04C6 44A8                   JMP    CBBAT1        ;ELSE IT'S CLEAR BATCH
04C8 C7             IOABK2: MOV    A,PSW         ;SEE IF LEGAL CHAR
04C9 D2CE                   JB6    IOERR         ;IF AUX CARRY, NOT LEGAL CHAR
04CB FD                     MOV    A,R5          ;ELSE GO TO ADD TO KEYBOARD
                    ;       JMP    FINNCH        ;ONLY NEED WITH KLUDGE I/O
04CC 4471                   JMP    FINN01        ;ELSE, GO TO FINN01
04CE BCC8           IOERR:  MOV    R4,#0C8H      ;THIS IS FOR BAD CMD OR DATA
04D0 E490           ITSK1J: JMP    IOTSK0
04D2 B91F           IOWRBJ: MOV    R1,#R7FORG             ;GET COPY OF TEST SIGNAL
04D4 64F0                   JMP    IOWRB
04D6 44E4           IOAKB1: JMP    FNCLK5
                    ;*****************************************************************
                    ;
                    ;      CHKTOT--S/R WHICH CHECKS THAT DREG + AREG = TOTAL.  IT CHECKS THAT
                    ;              THE TWO COPIES OF EACH ARE =, THEN MOVES DREG TO TDREG1&2,
                    ;              TOTAL TO BTEMP1&2, AREG TO TAREG1&2 AND BTEMP (BAM1)
                    ;              THEN IT ADDS TDREG1 AND BTEMP1 (COPY OF
                    ;              AREG) TO BTEMP1 AND COMPARES BTEMP WHICH HAS A
                    ;              COPY OF TOTAL IN BAM2.  THIS LEAVES A COPY OF TOTAL IN BTEMP.
```

BAD ORIGINAL

```
LOC  OBJ         LINE        SOURCE STATEMENT

                       ;              (BOTH BAMS)  FOR LATER USE
                       ;
                       ;************************************************************************
                                 ;
04D8 B900              CHKTOT: MOV     R1,#BDREG
04DA 7415                      CALL    COMPBA
04DC B878                      MOV     R0,#BTDREG
04DE 347E                      CALL    MOVR00
04E0 B920                      MOV     R1,#BTOTAL
04E2 7415                      CALL    COMPBA
04E4 B850                      MOV     R0,#BTEMP
04E6 347E                      CALL    MOVR00
04E8 B918                      MOV     R1,#BAREG
04EA 7415                      CALL    COMPBA
04EC B868                      MOV     R0,#BTAREG
04EE 347E                      CALL    MOVR00
04F0 B850                      MOV     R0,#BTEMP
04F2 3475                      CALL    MOVR01
04F4 B970                      MOV     R1,#BTDREG
04F6 B850                      MOV     R0,#BTEMP
04F8 85                        CLR     F0              ;READY TO ADD TDREG AND TAREG INTO
04F9 74A2                      CALL    ARRNC           ;BTEMP
04FB B950                      MOV     R1,#BTEMP
04FD 74B6                      CALL    COMPBK          ;SEE IF EQUAL TO COPY OF TOTAL
04FF B900                      MOV     R1,#BDREG       ;MOVE THE PERMANENT REGS TO TTEMP LOCATIONS
0501 B850                      MOV     R0,#BTTDR
0503 3478                      CALL    MOVR25
0505 83                        RET
                       ;*******************************************************************
                       ;
                       ;       XCHAPT--THIS SUBROUTINE XCHANGES R4 AND STEPIN(X) AND R5 AND
                       ;               STEPTK(X)
                       ;
                       ;************************************************************************
                               ;
0506 FB                XCHAPT: MOV     A,R3            ;GET PRTSTA  -
0507 5303                      ANL     A,#003H         ;EXPECTS R3 IN A
0509 4330                      ORL     A,#STEPIN       ;FORM ADDR FOR STEPIN(X)
050B A8                        MOV     R0,A
050C 433C                      ORL     A,#STEPTK       ;GET ADDR FOR STEPTK(X)
050E A9                        MOV     R1,A
050F FC                        MOV     A,R4            ;GET R4 READY FOR XCH
0510 20                        XCH     A,@R0           ;DO IT
0511 AC                        MOV     R4,A            ;COMPLETE XCHG
0512 FD                        MOV     A,R5
0513 21                        XCH     A,@R1
0514 AD                        MOV     R5,A
0515 FB                        MOV     A,R3            ;GET R3 BACK IN THE ACCUM
0516 83                        RET
                       ;*******************************************************************
                       ;
                       ;       PRINT--ROUTINE WHICH INITIALIZES STEPPERS, POSITIONS THEM, UPDATES
                       ;               THE BAM, AND GENERALLY CONTROLS THE PRINT CYCLE
                       ;               IT USES R3 TO STORE THE PRTSTA (PRINTER STATUS)--WHICH
                       ;               STEPPER, ENV OR TAPE, WHAT PART OF TASK BEING DONE,ETC
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                  1485 ;                  IT STORES IN R4 STEPIN(X), LSN = FINAL CODE (FROM KEYBD REG)
                  1486 ;                  MSN = CURRENT COMMAND TO STEPPER (X)
                  1487 ;                  IT STORES IN R5 STEPTK(X), LSN HAS NEXT POSITION EXPECTED,
                  1488 ;                  AND MSN HAS INFO ABOUT IF DONE POSITIONING, STEPPING CW OR
                  1489 ;                  CCW AND HOW MANY TIMES TRIED TO POSITION AND IF EXPECTING
                  1490 ;                  A 1/2 STEP VERIFY (-1)
                  1491 ;
                  1492 ;*************************************************************************
                  1493           ;
                  1494           ;
0517 FB           1495 PRINT:    MOV     A,R3            ;SET UP TO KEYSTA
0518 929E         1496           JB4     PTSK3X          ;IF P.O. MODE, SOFT FAULT
051A 430E         1497           ORL     A,#00EH         ;MAKE KEYBOARD FULL
051C A0           1498           MOV     @R0,A           ;STORE IN KEYSTA
051D 14FC         1499           CALL    DISFRG
051F B835         1500           MOV     R0,#PRTSTA      ;GET PRTSTA
0521 B938         1501           MOV     R1,#038H        ;WANT TO SEE IF OUT OF MONEY
0523 F1           1502           MOV     A,@R1
0524 1278         1503           JB0     PRTFXY          ;JUMP IF YES
0526 B91F         1504           MOV     R1,#R7FORG      ;FOR POSSIBLE USE IN PRINT1
0528 F0           1505           MOV     A,@R0
0529 AB           1506           MOV     R3,A            ;INTO A
052A D287         1507           JB6     PRINT1          ;IF DONE POSITIONING, DON'T NEED TO
                  1508                                   ;BRING IN STEPIN(X) AND STEPTK(X)
052C FA           1509           MOV     A,R2            ;MUST MAKE SURE NOT CLUTCH OR HOME
052D 37           1510           CPL     A               ;WAKEUP
052E 3234         1511           JB1     PRINT3          ;JUMP IF NOT
0530 F1           1512           MOV     A,@R1           ;FOR FAULTC
0531 37           1513           CPL     A               ;FOR EASE OF NEXT TEST
0532 64DD         1514           JMP     FAULTC          ;JUMP TO SEE HOME FAULT OR CLUTCH HALT
0534 B4A5         1515 PRINT3:   CALL    XCHAPT          ;BRING IN STEPIN(X) TO R4 AND STEPTK
                  1516                                   ;(X) TO R5
0536 B274         1517           JB5     PTSK2           ;DO JUMPS TO APPROPRIATE PTSKS
0538 9248         1518           JB4     PTSK1
                  1519           ;
                  1520           ;THIS IS BEGINNING OF PRINT WHEN JUST GET STATE OF NEW PRINT TASK
                  1521           ;AND DO PRELIMINARY CHECKING
                  1522           ;
                  1523 PTSK0:                            ;TURN OFF KEYBD AND I/O STUFF
053A 2301         1524           MOV     A,#INHKBI       ;THIS SAYS LOOK ONLY AT PRINTER
053C 74CE         1525           CALL    BAKRED
053E B82E         1526           MOV     R0,#REGSW
0540 F0           1527           MOV     A,@R0
0541 53C0         1528           ANL     A,#0C0H         ;GET PRTSTA INITLED WITH TAP/ENV
                  1529                                   ;AND REPEAT
0543 47           1530           SWAP    A               ;REPEAT IS BIT 3 AND ENV IS BIT 2
                  1531                                   ;THUS TAPE IS NOT BIT 2
0544 4B           1532           ORL     A,R3            ;SAVE INITING BIT IF EXISTS
0545 0310         1533 PTSK05:   ADD     A,#010H         ;CHANGE TO PRINTER TASK 1
0547 AB           1534           MOV     R3,A
                  1535           ;.
                  1536           ;PTSK1 GETS INITIAL POSITION OF STEPPERS STARTED
                  1537           ;
0548 34D1         1538 PTSK1:    CALL    SND
054A FB           1539           MOV     A,R3            ;SEE IF DONE WITH ALL FOUR
```

BAD ORIGINAL

```
LOC  OBJ         LINE       SOURCE STATEMENT

054B 37          1540        CPL     A
054C 1254        1541        JB0     PRTFA0          ;NO
054E 3254        1542        JB1     PRTFA0          ;NO
0550 FB          1543        MOV     A,R3            ;YES, GO TO NEXT TASK
0551 0310        1544        ADD     A,#010H
0553 AB          1545        MOV     R3,A
                 1546        ;
                 1547        ;PRTFA IS THE END OF TASKS 0 THRU 3
                 1548        ;IT DOES THE XCHG OF R4 AND R5 AND STEPIN(X) AND STEPTK(X)
                 1549        ;GETS THE NEXT STEPPER INTO PRTSTA
                 1550        ;
0554 B40E        1551 PRTFA0: CALL   XCHAPT
0556 3268        1552        JB1     PRTFA3          ;MUST SEE IF DOING LAST SINCE MUST
                 1553                                ;INCR LAST STEPPER TO 0
0558 17          1554 PRTFA5: INC    A               ;R3 IS IN A
0559 B835        1555 PRTFA7: MOV    R0,#PRTSTA
055B A0          1556 PRTFA8: MOV    @R0,A           ;STORE PRTSTA BACK
055C F268        1557        JB7     PRTFA6          ;JUMP IF INITING
055E E479        1558 PRTFAA: JMP    MAINR2
0560 1264        1559 PRTFA3: JB0    PRTFA4
0562 A458        1560        JMP     PRTFA5
0564 53FC        1561 PRTFA4: ANL    A,#0FCH         ;TO START AGAIN WITH ZERO STEPPER
0566 A459        1562        JMP     PRTFA7
0568 37          1563 PRTFA6: CPL    A               ;FOR EASE OF TESTING
0569 D25E        1564        JB6     PRTFAA          ;JUMP IF NOT DONE STEPPING
056B 34F8        1565 PRTFXX: CALL   TOFFSS          ;DONE INITING. TURN OFF STEPPERS
056D A0          1566        MOV     @R0,A           ;AND SOLEN(RETS WITH 0 IN A), MAKE PRTSTA = 0
056E 44E4        1567        JMP     FNCLKS          ;GO TO CLEAR KEYBOARD (AND MAKE KEYBD NOT FULL)
0570 239C        1568 PRTFXY: MOV    A,#09CH         ;TURN OFF ALL PRINTER BITS,SINCE OUT OF MONEY
0572 E476        1569        JMP     MAINR
                 1570 ;PTSK08:      MOV     A,#004H         ;P.O. MODE AND TRYING TO PRINT
                 1571 ;      JMP     FAULT0          ;ERROR
                 1572        ;
                 1573        ;THIS IS  CONTINUATION OF FINDING PRESENT POSITION
                 1574        ;OR POSITIONING   IF TASK 2,
                 1575        ;ITLOOKS FOR A VALID VERIFY CODE AND ACCEPTS THAT AS PRESENT
                 1576        ;POSITION   IF CAN'T FIND JUST CONTINUES TO TRY.   IF TASK 3
                 1577        ;AND CAN'T POSITION, FAULT, UNLESS INITING, AND THEN RETRYS.
                 1578        ;
0574 D439        1579 PTSK2: CALL    VER             ;THIS TURNS OFF
                 1580                                ;TIMER WAKEUP BIT AND RESTARTS
                 1581                                ;PRINTER TIMER COUNTER AND
                 1582                                ;CHECKS POSITION AND SETS BITS 567
                 1583                                ;IN STEPTK ACCORDING TO TABLE BELOW
0576 B680        1584        JF0     PTSK28          ;JUMP IF FOUND NEXT POSITION
0578 FD          1585        MOV     A,R5            ;ELSE SEE IF 2ND TIME TRY
0579 D29B        1586        JB6     PTSK32          ;JUMP IF YES
057B 4340        1587        ORL     A,#STPTRY       ;ELSE MAKE 2ND TIME TRY = 1
057D AD          1588        MOV     R5,A            ;PUT BACK IN R5
057E A454        1589        JMP     PRTFA0          ;GO TO END OF POSITIONING
0580 FB          1590 PTSK28: MOV    A,R3            ;SEE IF PTASK 2 OR 3
0581 9282        1591        JB4     PTSK38          ;JUMP IF 3
0583 37          1592 PTSK29: CPL    A               ;SEE IF LAST STEPPER
0584 1254        1593        JB0     PRTFA0          ;GO TO PRINT FIN IF NOT
0586 3254        1594        JB1     PRTFA0
```

```
LOC  OBJ      LINE        SOURCE STATEMENT

                          ;
                          ;THIS LAST STEPPER IS POSITIONED, ARE THE OTHERS?
                          ;FOLLOWING IS DECODE OF BITS 567 OF STEPTK
                          ;       765   DESCRIPTION
                          ;       000   JUST DONE NSTEP, OR NOT POSITIONING, OR JUST STARTING
                          ;       001   DONE WITH A FULL STEP
                          ;       010   DOING 2ND TIME TRY FOR FULL POSITION
                          ;       011   DONE POSITIONING
                          ;       100   DOING HALF STEP
                          ;       101   NO MEANING
                          ;       110   NO MEANING
                          ;       111   NO MEANING
                          ;
0588 B93E                 MOV     R1,#STPTK2      ;INIT TO STEPTK +2
058A BE03                 MOV     R6,#003H        ;ONLY HAVE TO CHECK 3
058C F1           PTSK21: MOV     A,@R1           ;GET PTSKTK(X)
058D 37                   CPL     A               ;FOR EASE OF TESTING
058E B254                 JB5     PRTFA0
0590 D254                 JB6     PRTFA0          ;JUMP OUT IF NOT DONE
0592 C9           PTK211: DEC     R1              ;ELSE DECREMENT POINTER
0593 E88C                 DJNZ    R6,PTSK21       ;DONE ALL?
0595 B406                 CALL    XCHAPT          ;YES, SEE IF INITING, OR WHICH TASK
0597 9201                 JB4     PTSK3A          ;TASK3, MUST CLEANUP
0599 C4A9                 JMP     PTSK2P          ;TASK2, JUMP
                          ;
                          ;THIS IS PRINTER TASK 3 CODE, MOST OF IT IS IN COMMON WITH TASK 2
                          ;
059B FB           PTSK32: MOV     A,R3            ;COULDN'T POSITION,
059C F2A0                 JB7     PTSK33          ;JUMP IF INITING
059E 64EC         PTSK3X: JMP     FAULT1          ;ELSE SOFT FAULT
05A0 E437         PTSK33: JMP     INITPW          ;TRY AGAIN INITING
                  ;MOV    A,#CHGDIR       ;TIRED 2 TIMES AND COULDN'T
                  ;XRL    A,R5            ;STEP BACK AND RETRY
                  ;MOV    R5,A            ;THIS SWITCHES DIRECTION
                  ;CALL   NSTEP
                  ;CALL   NSTEP           ;NSTEP RETURNS WITH R5 IN A
                  ;XRL    A,#CHGDIR       ;SWITCH BACK TO ORIGINAL DIR
                  ;MOV    R5,A
                  ;PTSK3K:        CALL    SND             ;SEND OUT NEW COMMAND
                  ;JMP    PRTFA0
05A2 FD           PTSK38: MOV     A,R5            ;POSITIONED, ARE WE TO FINAL POSIT
05A3 37                   CPL     A               ;FOR EASE OF TESTING BIT 5 & 6
05A4 D2AB                 JB6     PTSK39
05A6 B2AB                 JB5     PTSK39          ;JUMP IF NO
05A8 FB                   MOV     A,R3            ;GET READY FOR PTSK29
05A9 A493                 JMP     PTSK29
05AB D474         PTSK39: CALL    NSTEP           ;NO, GET NEXT COMMAND
05AD 34D1                 CALL    SND             ;SEND OUT NEW COMMAND
05AF A454                 JMP     PRTFA0
05B1 27           PTSK3A: CLR     A               ;SINCE TO FINAL POSITION
05B2 3A                   OUTL    P2,A            ;TURN ON SOLENOID
05B3 34AC                 CALL    TOPTIC          ;TURNS OFF PRCTR AND TIMER WAKEUP
05B5 C4DA                 JMP     PTK20D          ;JUMP
                          ;
                          ;THIS IS AFTER POSITIONING, TASK 4 OR 5, FIRST CHECK THAT NO TIMER
```

BAD ORIGINAL

```
LOC  OBJ         LINE        SOURCE STATEMENT

                 1650        ;WAKEUP (IF YES, HOME DIDN'T HAPPEN WITHIN 80 MSECS OF CLUTCH AND
                 1651        ;IS A FAULT).  THEN CHECK NOT TAPE OR ENV WAKEUP (JUST THOSE SIGNALS
                 1652        ;GOING AWAY, NO PROBLEM).  THEN TURN OFF PHOCL BIT.  SEE IF HOME = 1
                 1653        ;IF NOT, SEE IF CLUTCH  IF YES, START TIMER AND RETURN TO MAIN
                 1654        ;IF NOT CLUTCH, IF NOT TASK 5, JUST RETURN TO MAIN. ELSE BACK TO HOME,
                 1655        ;UPDATE BATCH STUFF AND COUNT AND END PRINT CYCLE AND RETURN TO MAIN
                 1656        ;IF HOME HAS LEFT, TURN OFF THE STEPPERS AND CLEAR THE PRINTER TIMER
                 1657        ;UPDATE DR AND AR AND CHECK RESULTS OF MOVES, AND CHANGE TO TASK 5 AND
                 1658        ;RETURN TO MAIN
                 1659        ;
05B7 FA          1660 PRINT1: MOV   A,R2          ;GET R2
05B8 B2EC        1661        JB5   CHFAUL        ;JUMP IF TIMER WAKEUP
05BA D2D1        1662        JB6   PTSK4E        ;JUMP IF TAPE OR ENV WAKEUP
05BC 53FD        1663        ANL   A,#NPHOCL     ;TURN OFF PHOCL BIT
05BE 65          1664        STOP  TCNT          ;WHILE CHANGING R2
05BF AA          1665        MOV   R2,A
05C0 55          1666        STRT  T
05C1 F1          1667        MOV   A,@R1         ;SET UP TO R7FORG
05C2 92E8        1668        JB4   PTSK41        ;JUMP IF LEFT HOME
05C4 B2E0        1669        JB5   PTSK4T        ;JUMP IF CLUTCH
05C6 FB          1670        MOV   A,R3          ;SEE IF TASK 4 OR 5
05C7 92CB        1671        JB4   PRINT8        ;RETURN TO MAIN IF TASK 4
05C9 E479        1672 PRINT7: JMP   MAINR2        ;IT'S TASK 4
05CB B2CF        1673 PRINT8: JB5   PRINT9        ;IF TASK 7, DON'T TURN OFF STEPPERS
05CD 34F8        1674        CALL  TOFFSS        ;SINCE ANOTHER ENV COMING
05CF C41F        1675 PRINT9: JMP   PTSK51        ;GO TO END OF PRINT
                 1676        ;
                 1677        ;
                 1678        ;HAVE ENV/TAPE WAKEUP, IF NEW ENVELOPE, GO TO TASK 6 (IF NOW TASK 4)
                 1679        ;OR TASK 7 (IF NOW TASK 5), OTHER WISE JUST TAPE OR ENV GOING AWAY,
                 1680        ;TURN OF WAKEUP AND GO BACK TO MAIN
                 1681        ;
05D1 B92E        1682 PTSK4E: MOV   R1,#REGSW
05D3 F1          1683        MOV   A,@R1         ;GET TAPE AND ENV
05D4 D2DA        1684        JB6   PTSK4F        ;IT'S ENVELOPE, JUMP
05D6 23BF        1685 PTSK4H: MOV   A,#NPTAEN     ;TO TURN OF TAPE/ENV WAKEUP
05D8 E476        1686        JMP   MAINE
05DA F0          1687 PTSK4F: MOV   A,@R0         ;SET UP TO PRTSTA
05DB 4320        1688        ORL   A,#020H       ;CHANGE TO TASK 6 OR 7
05DD A0          1689        MOV   @R0,A         ;PUT BACK INTO PRTSTA
05DE A4D6        1690        JMP   PTSK4H
                 1691        ;
                 1692        ;FOUND CLUTCH. IF DR NOT ALREADY UPDATED (PTASK = 5 OR 7)
                 1693        ;START TIMER FOR 1ST 112 MSECS
                 1694        ;
05E0 FB          1695 PTSK4T: MOV   A,R3          ;SEE IF ALREADY GOT HOME
05E1 925E        1696        JB4   PRTFAA        ;JUMP IF YES
05E3 53FC        1697        ANL   A,#0FCH       ;CLEAR OUT TWO STEPPER BITS FOR 224 MSEC TIMER
05E5 A0          1698        MOV   @R0,A         ;STORE INTO PRTSTA
05E6 23FF        1699        MOV   A,#0FFH       ;GIVE 100 MSECS FOR HOME
05E8 34AC        1700        CALL  TOPTIC        ; CHANGE PRCTR
05EA A4FE        1701        JMP   PTSK42
05EC FB          1702 CHFAUL: MOV   A,R3          ;SEE IF 1ST TIME THROUGH  (1ST 112 MS)
05ED 12F4        1703        JB0   HFAULT        ;JUMP IF THRU 2ND
05EF 18          1704        INC   @R0           ;ELSE INCR PRTSTA
```

BAD ORIGINAL

```
LOC  OBJ      LINE          SOURCE STATEMENT

05F0 23FF     1705          MOV    A,#0FFH          ;AND START 2ND 112 MSEC WAIT
05F2 A4F9     1706          JMP    PTSK4P
05F4 2302     1707 HFAULT:  MOV    A,#002H          ;THIS IS NO HOME FAULT
05F6 64E1     1708          JMP    FAULT0
              1709          ;
              1710          ;THIS IS CODE FOR WHEN LEFT HOME OR GOT CLUTCH —
              1711          ;UP DATE AR AND DR , THEN GO TO TASK 5 OR 7
              1712          ;
05F8 27       1713 PTSK41:  CLR    A
05F9 34AC     1714 PTSK4P:  CALL   TOPTIC           ;THIS CLEARS PRCTR AND DOES STRT I
05FB FB       1715          MOV    A,R3             ;ALREADY DONE TASK 4?
05FC 925E     1716          JB4    PRTFAA           ;JUMP IF YES
              1717 PTSK42:;           MOV    A,R2             ;IF POWERLOSS, JUST STOP
              1718 ;        JB7    $
05FE D433     1719          CALL   MVTDDR           ;MOVE NEW DR TO DR FROM TDR
0600 B900     1720          MOV    R1,#BTAREG
0602 B810     1721          MOV    R0,$BAREG
0604 347E     1722          CALL   MOVRBB           ;MOVE NEW AR TO AR FROM TAR
              1723          ;
              1724          ;NOW UPDATE COUNT, BATCH COUNT
              1725          ;AND BATCH TOTAL
              1726          ;
0606 85       1727          CLR    F0
0607 B830     1728          MOV    R0,#BBATL        ;NOW ADD KB TO BATCH COUNT
0609 74A0     1729          CALL   ARKBNC
060B B9A0     1730          MOV    R1,#BRGTHO       ;THIS REGISTER HAS 1000 IN IT FOR
              1731                                   ;ADDING 1 000 TO BATCH COUNT
060D B840     1732          MOV    R0.#BBACT
060F 74A2     1733          CALL   ARRNC
0611 B9A3     1734          MOV    R1,#BRGONE       ;THIS REGISTER HAS ONLY 1 IN IT
0613 B850     1735          MOV    R0.#BCOUNT       ;THIS IS ADDR OF THE COUNT REGISTER
0615 74A2     1736          CALL   ARRNC
0617 B930     1737          MOV    R1,#BBATL        ;COPY BOTH THE BATCH REGISTERS AND THE
0619 B830     1738          MOV    R0,#BBATL        ;COUNT REGISTER TO RAM 2
061B 3478     1739          CALL   MOVRBS           ;WRITE TO BOTH RAMS
061D C4DA     1740          JMP    PTK20D           ;CHANGE TO TASK-5 (OR 7)
              1741          ;
              1742          ;THE END OF THE PRINTER TASK, ITS CLEARS PRSTA IF NOT DOING
              1743          ;CONSECUTIVE ENVS. CHECKS IF HAVE
              1744          ;
              1745 PTSK51:;           MOV    R0,#PRTSTA
061F FB       1746          MOV    A,R3             ;CHECK TAPE AND REPEAT
0620 B22A     1747          JB5    PTSK57           ;1ST SEEING IF DOING CONSEC ENVS
0622 B000     1748          MOV    @R0,#000H        ;NO, CLEAR PRTSTA
0624 522D     1749          JB2    PTSK55           ;IT'S ENVELOPE, JUMP
0626 722D     1750          JB3    PTSK55           ;IT'S TAPE AND REPEAT, JUMP
0628 44E4     1751          JMP    FNCLK5           ;THIS CLEARS KEYBD REG AND REINITS
              1752                                   ;TDREG AND TAREG
062A 534C     1753 PTSK57:  ANL    A,#04CH          ;KEEP TAPE/ENV, REPEAT AND TASK 4
062C A0       1754          MOV    @R0,A            ;STORE IN PRTSTA
062D B834     1755 PTSK55:  MOV    R0,#KEYSTA       ;NEED TO FIX UP R3 WITH KEYSTA
062F F0       1756          MOV    A,@R0
0630 AB       1757          MOV    R3,A
0631 6451     1758          JMP    AKREG4
              1759 ;**********************************************************************
```

BAD ORIGINAL

```
LOC  OBJ         SOURCE STATEMENT

                 ;
                 ;       REGMOV--S/R WHICH MOV CONTENTS OF TEMP DREG TO PERMANENT
                 ;               DREG (BOTH BAMS)
                 ;
                 ;**************************************************************
                         ;
0633 B97A        MVTDDP: MOV    R1,#BTDREG
0635 B800                MOV    R0,#BDREG
0637 247E                JMP    MOVR00
                         ;
                 ;**************************************************************
                 ;
                 ;       VER--THIS S/R GETS THE VERIFY CODE   IF THE VERIFY
                 ;               CODE MATCHES THE NEXT POSITION, THEN IT LOOKS TO SEE
                 ;               IF THE VERIFY CODE MATCHES THE FINAL CODE.  IN EITHER
                 ;               CASE, IF SETS F0 TO LET THE CALLING ROUTINE KNOW THAT
                 ;               THE VERIFY MATCHED.  IF THE STEPPER IS POSITIONED TO
                 ;               THE FINAL CODE, THEN THE DONE BIT IS SET AS WELL
                 ;
                 ;**************************************************************
                         ;
0639 85          VER:    CLR    F0                   ;F0=0 WILL SAY NO MATCH BETWEEN PRESENT
                                                     ;(STEPTK(X)) AND VERIFY
063A 34AA                CALL   TOPTI                ;THIS TURNS OFF THE TIMER TASK BIT
                                                     ;AND RESTARTS THE PRINTER TIMER
063C 74C4                CALL   DSENP6               ;PUTS PORT 2 IN THE RIGHT STATE
063E FB                  MOV    A,R3                 ;SET UP VERIFY ADDR
063F 5303                ANL    A,#003H              ;ONLY WANT STEPPER(X) INFO
0641 E7                  RL     A
0642 E7                  RL     A
0643 BF10                MOV    R7,#010H             ;CLEAR R7  WHERE STORE VER POS
                                                     ;USE THE ONE BIT TO TELL WHEN DONE
0645 3C          VERLP:  MOVD   P4,A                 ;SEND OUT ADDR TO VADR
0646 A8                  MOV    R0,A                 ;SAVE IN R0
0647 364A                JT0    VER1                 ;JUMP IF GET 0 BACK (INVERTED OUTPUT)
0649 1F                  INC    R7
064A FF          VER1:   MOV    A,R7                 ;DO RL OF R7
064B F253                JB7    VER2                 ;UNLESS DONE
064D E7                  RL     A
064E AF                  MOV    R7,A
064F F8                  MOV    A,R0                 ;GET BACK POINTER
0650 17                  INC    A                    ;INCR IT
0651 C445                JMP    VERLP
0653 FD          VER2:   MOV    A,R5                 ;SEE IF LOOKING FOR -1 VERIFY
0654 F26F                JB7    VER8                 ;JMP IF YES
0656 DF                  XRL    A,R7                 ;NO, COMPARE NEXT POSITION
                                                     ;WITH VERIFY POSITION
0657 530F                ANL    A,#CLRMSH
0659 C65C                JZ     VER5                 ;JUMP IF MATCH
065B 83                  RET                         ;ELSE RET
065C FD          VER5:   MOV    A,R5                 ;STEP DONE, SET BIT
065D 539F                ANL    A,#NDONEP            ;AFTER CLEARING TWO DONE BITS
065F 4320                ORL    A,#STPDON
0661 AD                  MOV    R5,A
0662 95                  CPL    F0                   ;FOUND A MATCH
```

BAD ORIGINAL

```
LOC  OBJ        LINE        SOURCE STATEMENT

0563 FC                     MOV     A,R4            ;SEE IF FINAL POSITION = VER POSIT
0564 DF                     XRL     A,R7
0565 530F                   ANL     A,#CLRMSN
0567 C66A                   JZ      VER3
0569 83                     RET                     ;RETURN IN NOT DONE POSITIONING
056A FD              VER3:  MOV     A,R5            ;GET READY TO SET DONE POSITIONING
056B 436A                   ORL     A,#POSDON
056D AD                     MOV     R5,A
056E 83                     RET
056F 539F            VER8:  ANL     A,#NDONEP       ;CLEAR TWO DONE BITS
0671 AD                     MOV     R5,A
0672 95                     CPL     F0              ;DONE POSITIONING 1/2 STEP
0673 83                     RET                     ;(PROBABLY)
                     ;*****************************************************************
                     ;
                     ;      NSTEP--THIS SUBROUTINE CHANGES THE STEPPER'S INSTRUCTION
                     ;             TO DO THE NEXT STEP.  IT MUST TAKE INTO ACCOUNT THE
                     ;             DIRECTION TO CHANGE THE STEPPER INST (MSN STEPIN(X)).
                     ;             THEN IT SEES IF THE LAST TIME CHANGE WAS FOR A 1/2
                     ;             POSITION.  IF YES, IT CHANGES THAT, IF NO, IT MUST INC
                     ;             OR DECR THE NEXT POSITION (LSN OF STEPTK(X)).
                     ;
                     ; ;*****************************************************************
                            ;
0674 85              NSTEP:  CLR     F0             ;NOT INITING
0675 FD              NSTEP1: MOV     A,R5           ;FIRST GET DIRECTION
0676 929D                   JB4     NSTEP8          ;JUMP IF INCREASE
0678 FC                     MOV     A,R4            ;IT'S DECREASE, WILL DO A RR, IF B4=1
0679 92A5                   JB4     NSTEP9          ;MUST JUMP
067B 53F0                   ANL     A,#CLRLSN       ;
067D 77                     RR      A               ;ELSE ROTATE RIGHT
067E AF              NSTEP0: MOV    R7,A            ;SAVE IN R7
067F FC              NSTEP1: MOV    A,R4            ;NOW PUT NEW COMMAND BACK TO R4
0680 530F                   ANL     A,#CLRMSN       ;ZERO OUT MSN
0682 4F                     ORL     A,R7            ;OR IN THE NEW COMMAND
0683 AC                     MOV     R4,A            ;PUT BACK
0684 B69E                   JF0     NSTEPE          ;JUMP IF INITING
0686 FD                     MOV     A,R5            ;GET R5 TO CHECK ON LAST TIME -1
0687 539F                   ANL     A,#NDONEP       ;TURN OFF STEP AND DONE POSIT BITS
0689 F28F                   JB7     NSTEP3          ;WAS EXPECTED, JUMP IF YES
068B 4380                   ORL     A,#STPM1        ;ELSE MAKE THE NEXTPOSITION -1
068D AD              NSTEP2: MOV    R5,A            ;STORE BACK IN R5
068E 83              NSTEPE: RET                    ;AND RETURN
                            ;
                            ;NEED TO INCR OR DECR NEXT POSITION
                            ; OR DEC IF (R5(4)=0)
                            ;
068F 9296            NSTEP3: JB4    NSTEP5          ;B4=1 SAYS INCREASE
0691 07                     DEC     A               ; THIS IS DECREASE
0692 537F            NSTEP6: ANL    A,#NSTPM1       ;NOW MAKE NEXT TIME -1 = 0
0694 C48D                   JMP     NSTEP2
0696 17              NSTEP5: INC    A               ;ELSE INCREASE A
0697 C492                   JMP     NSTEP6
0699 BF10            NSTEP7: MOV    R7,#010H        ;FOR RL WITH MSB = 1
069B C47E                   JMP     NSTEP1
```

BAD ORIGINAL

```
 LOC  OBJ        LINE        SOURCE STATEMENT

 069D FC         1870 NSTEP8: MOV     A,R4          ;THIS IS FOR INCREASE
 069E F299       1871         JB7     NSTEP7        ;JUMP IF RL GOES TO LSB
 06A0 53FE       1872         ANL     A,#CLRLSH
 06A2 E7         1873         RL      A             ;INCREASE MEANS RL
 06A3 C47E       1874         JMP     NSTEP0
 06A5 BF80       1875 NSTEP9: MOV     R7,#080H      ;FOR RR WITH LSB = 1
 06A7 C47F       1876         JMP     NSTEP1
                 1877 ;*************************************************************
                 1878 ;
                 1879 ;       PTSK2P--PART OF PRINT ROUTINE
                 1880 ;
                 1881 ;*****************************************************************
                 1882         ;
                 1883         ;MUST GET KEYBOARD REG INTO FINAL POSITION(X) AND SET UP DIRECTION
                 1884         ;WHICH DEPENDS ON DIFFERENCE BETWEEN FINAL AND PRESENT POSITION
                 1885         ;
 06A9 BF04       1886 PTSK2P: MOV     R7,#004H      ;GET READY TO MOVE KEYBD REG TO
                 1887                               ;STEPIN(X)
 06AB B960       1888         MOV     R1,#BKBREG
 06AD B834       1889         MOV     R0,#KEYSTA    ;SEE IF FCENTS OR NOT
 06AF F0         1890         MOV     A,@R0
 06B0 D2B3       1891         JB6     PTK204        ;JUMP IF FCENTS
 06B2 19         1892         INC     R1            ;ELSE READ FROM KBREG(1)
 06B3 B836       1893 PTK204: MOV     R0,#STEPIN
 06B5 743F       1894 PTSK25: CALL    REDBM1
 06B7 30         1895         XCHD    A,@R0         ;MOVE KEYBD(X) TO STEPIN(Y)
 06B8 18         1896         INC     R0
 06B9 EFB5       1897         DJNZ    R7,PTSK25     ;DONE?
 06BB BB04       1898         MOV     R3,#004H      ;YES, NOW PICK INCR OR DEC
                 1899                               ;NOTE USING R3(ALL OTHER REGS
                 1900                               ;BUSY), WILL REESTABLISH AT END
 06BD B485       1901 PTSK26: CALL    XCHAPT        ;GET STEPIN(X) AND STEPTK(X)
 06BF FD         1902         MOV     A,R5          ;GET STEPTK(X)
 06C0 530F       1903         ANL     A,#CLRMSH     ; - PRESENT (STEPTK(X))
 06C2 AE         1904         MOV     R6,A          ;SAVE TEMPORARILY IN R6
 06C3 FC         1905         MOV     A,R4          ;GET STEPIN(X)
 06C4 530F       1906         ANL     A,#CLRMSH
 06C6 37         1907         CPL     A
 06C7 6E         1908         ADD     A,R6          ;THIS IS COMPLEMENT OF RESULT
 06C8 F2CD       1909         JB7     PTK200        ;SO IF JUMP, RESULT (+) NEED TO INC
 06CA FE         1910         MOV     A,R6          ;GET STEPTK(X) WITH MSN = 0
 06CB C4D3       1911         JMP     PTK201
 06CD 37         1912 PTK200: CPL     A             ;SEE IF NO CHANGE (RESULT COMPL = 0)
 06CE C6D4       1913         JZ      PTK202        ;JUMP IF NO CHANGE
 06D0 FE         1914         MOV     A,R6          ;THIS IS INCREASE
                 1915                               ;R6 IS STEPTK(X) WITH MSN = 0
 06D1 4310       1916         ORL     A,#010H       ; SO INCREASE
 06D3 AD         1917 PTK201: MOV     R5,A          ;PUT STEPTK IN R5 FOR NSTEP
 06D4 B485       1918 PTK202: CALL    XCHAPT        ;RESTORE STEPIN(X) AND STEPTK(X)
 06D6 1B         1919         INC     R3            ;INCR POINTER
 06D7 FB         1920         MOV     A,R3          ;SEE IF DONE YET
 06D8 52BD       1921         JB2     PTSK26        ;JUMP IF NOT
 06DA B835       1922 PTK20D: MOV     R0,#PRTSTA    ;RESTABLISH PRTSTA IN A
 06DC F0         1923         MOV     A,@R0         ;FOR PRTFA3
 06DD 0310       1924         ADD     A,#010H       ;GOES TO TASK 3
```

BAD ORIGINAL

0019515


```
LOC  OBJ        LINE       SOURCE STATEMENT

06DF A45B                  JMP     PRTFA8          ;AND JUMP
                           ;
                ;*******************************************************************
                ;
                ;
                ;   INIT -- THIS IS INITIALIZATION FOR THE FRIDEN ELECTRONIC POSTAL
                ;                  METER MICRO CODE.  IT INITS VARIOUS REGISTERS,
                ;                  TURNS OFF DISPLAY AND STEPPERS.  IT STARTS THE TIMER AND WAITS
                ;                  100 MSEC FOR SYSCLR TO GO AWAY, THEN STARTS DISPLAY THEN
                ;                  AFTER FOREGROUND HAS MADE THREE PASSES TO UPDATE VARIOUS
                ;                  REGISTERS, THE FOREGROUND IS TURNED OFF, AND THE RAM IS
                ;                  CHECK OUT.  THEN IT IS CHECK TO SEE IF THE METER IS
                ;                  ON A BASE OR NOT.  IF ON A BASE, THE STEPPERS ARE STEPPED
                ;                  TO NINE (WITH NO VERIFY, BUT FINISHING WITH A KNOWN COMMAND)
                ;                  THEN RETURNED TO ZERO UNDER NORMAL PROGRAM CONTROL.  FINALLY,
                ;                  THE DISPLAY IS AGAIN REENABLED AND CONTROL
                ;                  GOES TO THE MAIN BACKGROUND LOOP AND WAITS FOR A TASK.
                ;
                ;
                ;*******************************************************************
                           ;
                           ;THIS IS BEGINNING OF INITIALIZATION CODE
                           ;
06E1 2340       INIT:      MOV     A,#SOLEN        ;FIRST TURN OFF SOLENOID
06E3 3A                    OUTL    P2,A
06E4 27                    CLR     A               ;NOW CLEAR OUT ALL RAM
06E5 B83F                  MOV     R0,#03FH
06E7 A0         INIT1:     MOV     @R0,A
06E8 E8E7                  DJNZ    R0,INIT1
06EA D5                    SEL     RB1                    ;FIRST FIX UP FOREGROUND REGISTERS
06EB B920                  MOV     R1,#SAVACC      ;INI FOR CALLS TO INTERRUPT ROUTINES
06ED B82A                  MOV     R0,#REG07       ;INI FOR START OF DISPLAY & DEBOUNCING
06EF BAA0                  MOV     R2,#0A0H        ;INI CHARACTER POINTER TO NOT DISPLAY
06F1 C5                    SEL     RB0             ;GO TO BANK 0 REGISTERS
06F2 B936                  MOV     R1,#PRCTR       ;GET ADDR OF PRCTR TO INITIALIZE IT
06F4 B1FF                  MOV     @R1,#MINONE
06F6 23FA                  MOV     A,#TIMCNT       ;TO 255*400 MICROSECS OR APPROX
                                                   ;100 MILLISECS OF WAIT UNTIL START TO
                                                   ;ACCESS RAM, 8243S, ETC.
06F8 62                    MOV     T,A             ;MOVE -1 TO TIMER COUNTER
06F9 25                    EN      TCNTI           ;ENABLE THE TIMER INTERRUPT TO START
                                                   ;THE FAULT FF ONESHOTS & TO COUNT DOWN
                                                   ;PRCTR SO THAT INIT ROUTINE CAN KNOW
                                                   ;THAT SYSCLR IS GONE (HOPEFULLY)
06FA 55                    STRT    T               ;START THE TIMER
                                                   ;MAKE NON ZERO WHEN PRCTR IS COUNTED DOWN
06FB FA         WAIT:      MOV     A,R2            ;GET NEW COPY OF BAKSTA
06FC C6FB                  JZ      WAIT            ;WHEN FOREGROUND COUNTS DOWN
                                                   ;PRCTR IT WILL SET A BIT IN BAKSTA
06FE 05                    EN      I               ;ENABLE THE POWERLOSS INTERRUPT
                           ;
06FF B920                  MOV     R1,#BFAULT      ;SEE IF METER FAULTED
0701 343C                  CALL    REDBMB
0703 65                    STOP    TCNT            ;IF NOT ZERO, FAULT, SHOULDN'T EVEN
0704 9604                  JNZ     $               ;BE EXECUTING, JUST STOP WITH TCNT OFF
```

BAD ORIGINAL

```
 LOC  OBJ         LINE        SOURCE STATEMENT

 0706 55          1980        STRT    T               ;FAULT DOES THAT
 0707 740E        1981        CALL    COMPBN          ;R1 POINTS TO BMAXPO, SEE IF BOTH =
 0709 AC          1982        MOV     R4,A            ;GETTING METTYPE READY
 070A 740E        1983        CALL    COMPBN          ;GETTING METYPE BITS
 070C 65          1984        STOP    TCNT            ;MAKE SURE NOT ZERO
 070D C60D        1985        JZ      $               ;JUMP IF YES
 070F 55          1986        STRT    T
 0710 B904        1987        MOV     R1,#004H        ;GET READY TO MOVE BMAXPO TO LSN
 0712 31          1988        XCHD    A,@R1
 0713 B917        1989        MOV     R1,#METYPE
 0715 A1          1990        MOV     @R1,A           ;STORE INTO METTYPE
 0716 5340        1991        ANL     A,#040H         ;NOW GET FCENTS
 0718 B92C        1992        MOV     R1,#REG89       ;FIRST STORE INTO REG89
 071A A1          1993        MOV     @R1,A
 071B 430A        1994        ORL     A,#00AH         ;OR WITH NO KEY PUSHED
 071D B934        1995        MOV     R1,#KEYSTA
 071F A1          1996        MOV     @R1,A           ;STORE IN KEYSTA
                  1997        ;NOW CHECK RAM CONTENTS FOR EQUAL CONTENTS IN EACH REGISTER
                  1998        ;
 0720 345E        1999        CALL    CLRKYR          ;CHECKS THAT TWO COPIES OF DR
                  2000                                ;ARE =, TWO COPIES OF AREG AND
                  2001                                ;TOTAL ARE LIKEWISE =, THEN MOVES
                  2002                                ;DREG TO TDREG, AREG TO TAREG
                  2003                                ;TOTAL TO TEMP AND CHECKS THAT TDREG
                  2004                                ;+ TAREG = COPY OF TOTAL
 0722 B9A2        2005        MOV     R1,#BRGTHO      ;SEE IF COUNT INCR REGISTERS ARE OK
 0724 BE0D        2006        MOV     R6,#00DH        ;CHECK MORE THAN A REGISTERS WORTH
 0726 741F        2007        CALL    CMPLOT          ;RETURNS WITH 0 IF NO ERROR
 0728 AA          2008        MOV     R2,A            ;CLEAR OUT R2
 0729 B836        2009        MOV     R0,#PRCTR       ;NOW LET FOREGROUND RUN FOR 100 MSECS
 072B B0FF        2010        MOV     @R0,#0FFH
 072D 74D5        2011        CALL    ENAFRG          ;ENABLE TO ALLOW SETTING OF REGSW
                  2012                                ;AND P.O. MODE, NORM, ETC
 072F FA          2013 WAIT1: MOV     A,R2            ;WAIT FOR FOREGROUND TO MAKE NON ZERO
 0730 37          2014        CPL     A               ;FOR EASE OF TESTING IF 100 MSEC
 0731 B22F        2015        JB5     WAIT1           ;HAS COME UP (8 PASSES OF FOREGND)
 0733 14FC        2016        CALL    DISFRG          ;TURN OFF DISPLAY WHILE DOING NEXT
 0735 BA10        2017        MOV     R2,#SWTSK       ;CLEAR TIMER BIT AND LEAVE REGTSK
 0737 B81F        2018 INITPW: MOV    R0,#R7FORG      ;SEE IF CONNECTED TO A METER BASE
 0739 F0          2019        MOV     A,@R0
 073A 723E        2020        JB3     INITPX          ;JUMP IF YES
 073C 44E4        2021        JMP     FNCLKS          ;THIS DOES CLEAR KBREG AND
                  2022                                ;GOES TO MAIN
 073E 923E        2023 INITPX: JB4    $               ;IF NOT AT HOME, JUST STOP
 0740 B240        2024        JB5     $               ;IF CLUTCH PENDING, JUST STOP
 0742 B935        2025        MOV     R1,#PPTSTA      ;GET READY TO INIT PRINTER STAT
 0744 B180        2026        MOV     @R1,#080H       ;THIS TELLS PRINTER ROUTINE INITING
 0746 BC09        2027        MOV     R4,#009H        ;GET READY TO STEP PRINTER TO 9 BY
 0748 BB08        2028        MOV     R3,#008H        ;FORCE, WILL SEND 30 INCR STEPS
 074A BD10        2029        MOV     R5,#010H        ;
 074C 34D1        2030 INITP0: CALL   SND             ;SEND STEP COMMAND AND RESTART TIMER
 074E 1B          2031        INC     R3              ;INCREMENT THE STEPPER POINTER
 074F FB          2032        MOV     A,R3            ;SEE IF HAVE DONE 4
 0750 37          2033        CPL     A               ;FOR EASE OF TESTING
 0751 125C        2034        JB0     INITP2
```

```
LOC  OBJ        LINE        SOURCE STATEMENT

0753 325C       2035          JB1    INITP2
0755 37         2036          CPL    A
0756 F261       2037          JB7    INITPF          ;SEE IF HAVE SENT 32 COMMANDS
0758 85         2038          CLR    F0
0759 95         2039          CPL    F0              ;THIS IS FOR USE BY NSTEPI
075A D475       2040          CALL   NSTEPI          ;GET NEXT COMMAND
075C FA         2041 INITP2:  MOV    A,R2            ;GET COPY OF R2 TO SEE IF TIMER
075D B24C       2042          JB5    INITP0          ;HAS TIMED OUT
075F E45C       2043          JMP    INITP2          ;ELSE LOOP
0761 B837       2044 INITPF:  MOV    R0,#STEPIN-1    ;INI THE INSTRUCTION TO THE STEPPERS
                2045                                 ;INI R0 TO STEPIN(0)
0763 FC         2046          MOV    A,R4            ;THIS SERVES TO INI BOTH
                2047                                 ;STEPTK AND STEPIN
                2048
0764 BF08       2049          MOV    R7,#008H        ;INI A LOOP COUNTER
0766 18         2050 INIT4:   INC    R0
0767 A0         2051          MOV    @R0,A           ;MOVE THE COMMAND TO STEPIN(X) AND STEPTK(X)
0768 EF66       2052          DJNZ   R7,INIT4        ;IT IS 029H
076A 9917       2053          MOV    P1,#METYPE      ;WANT TO SEE IF 3 OR 4 STEPPER METER
076C F1         2054          MOV    A,@R1
076D F274       2055          JB7    INITF           ;JUMP IF 4
076F 27         2056          CLR    A               ;ELSE MUST CLEAR PRESENT AND FINAL POSITION
0770 A0         2057          MOV    @R0,A
0771 B83B       2058          MOV    R0,#STEPIN+3
0773 A0         2059          MOV    @R0,A
0774 44E4       2060 INITF:   JMP    FNCLK5          ;CLEARS KEYBOARD AND DISPLAYS AND
                2061                                 ;GOES TO MAIN
                2062          ;
0776 65         2063 MAINR:   STOP   TCNT            ;STOP TCNT WHILE CHANGING BAKSTA
0777 5A         2064          ANL    A,R2            ;A HAS MASK IN IT TO DROP A BAKSTA BIT
0778 AA         2065          MOV    R2,A            ;PUT BACK IN BAKSTA
0779 74D5       2066 MAINR2:  CALL   ENAFRG
077B B834       2067          MOV    R0,#KEYSTA      ;GET KEYSTA TO R3 IN CASE KEYBD
077D F0         2068          MOV    A,@R0           ;OR REGDSP
077E AB         2069          MOV    R3,A
                2070          ;
                2071          ;THIS IS THE MAIN BAKGROUND LOOP WHICH DETERMINES WHICH BACK
                2072          ;GROUND TASK SHOULD BE JUMPED TO IF ANY.
                2073          ;
077F FA         2074 MAIN:    MOV    A,R2            ;GET NEW COPY OF BAKSTA
0780 F280       2075          JB7    $
0782 9292       2076          JB4    REGDSJ
0784 128A       2077          JB0    MAIN2           ;IF BIT 0 ON, NO KEYBD, I/O
0786 5294       2078          JB2    IORTSK
0788 7290       2079          JB3    KEYBDJ
078A 5362       2080 MAIN2:   ANL    A,#062H         ;ISOLATE THE PRINTER TASKS
078C C67F       2081          JZ     MAIN
078E A417       2082          JMP    PRINT
0790 4404       2083 KEYBDJ:  JMP    KEYBD
0792 8434       2084 REGDSJ:  JMP    REGDSP
                2085 ;********************************************************************
                2086 ;
                2087 ;        IORTSK--THE ROUTINE WHICH SERVICES I/O REQUESTS.  THE METER IS THE
                2088 ;                MASTER IN CONTROLLING A I/O TRANSFERS TO AND FROM THE
                2089 ;                UPI, EXCEPT THAT THE UPI TELLS THE METER SOMETHING IS TO
```

```
LOC  OBJ      LINE        SOURCE STATEMENT

              2090 ;              BE DONE BY PULLING /EXTINT1 LOW. WHICH CAUSES A I/O WAKEUP.
              2091 ;              THEN THE IORTSK ROUTINE READS UPI STATUS AND SENDS A COMMAND
              2092 ;              TO LOAD THE UPI'S DBB. THE IORTSK AGAIN READS THE STATUS UNTIL
              2093 ;              THE UPI HAS DONE THAT, THEN READS THE COMMAND, THEN AFTER
              2094 ;              DOING WHAT IS NECESSARY AND THEN READING STATUS AGAIN, EITHER
              2095 ;              THE DATA WORD IS READ, OR IT IS SEND TO THE UPI.  IF IT WAS A
              2096 ;              DATA WORD TO THE METER, THE METER IS DIRECTED TO DO WHATEVER
              2097 ;              IS NECESSARY WITH THE COMMAND AND DATA   THE COMMANDS WHICH
              2098 ;              THE METER IS ABLE TO DO ARE: 0-ILLEGAL. 1--COPY DATA WORD TO
              2099 ;              IOPTR. 2--ADD CHAR TO KEYBD REG (OR CLR KEYBOARD OR BATCH).
              2100 ;              3--READ BAM USING THE IOPTR (AND THEN INCR THE IOPTR).
              2101 ;              4--WRITE BAM USING THE IOPTR (AND THEN INCR THE IOPTR).
              2102 ;              THE LAST COMMAND IS ONLY POSSIBLE IF THE SIGNAL /TEST IS TRUE
              2103 ;              WHICH IS POSSIBLE ONLY WHEN THE METER IS UNCOVERED AT THE
              2104 ;              FACTORY.
              2105 ;
              2106 ;*******************************************************************
              2107          ;
0794 14FC     2108 IORTSK: CALL  DISFRG
0796 27       2109 IOTSKZ: CLR   A
0797 AC       2110         MOV   R4,A          ;CLEAR THE CURRENT COMMAND REGISTER
0798 85       2111 IOTSK0: CLR   F0            ;F0=1 SAYS COMMAND TO LOAD OR READ DBB
              2112                               ;HAS BEEN SENT
0799 55       2113 IOTSK1: STRT  T             ;FOR LOOPING
079A BB37     2114         MOV   R1,#IOPTR     ;FOR LATER USE
079C F1       2115         MOV   A,@R1
079D A8       2116         MOV   R0,A
079E A9       2117         MOV   R1,A          ;FOR THE READ AND WRITE BAM IOTSKS
079F 65       2118         STOP  TCNT          ;TO KEEP FOREG FROM CHANGING P6
07A0 2309     2119         MOV   A,#ENAIO      ;THIS WILL FIX-UP PORT 6 OF THE DISPLAY
07A2 74C6     2120         CALL  P6MOVD        ;8279 TO ENABLE THE I/O
07A4 8A38     2121         ORL   P2,#DPROG     ;THIS TURNS OFF BAM ENABLES TO BUSS
07A6 8901     2122 IOTSK5: ORL   P1,#001H      ;READY TO READ UPI STATUS
07A8 08       2123         INS   A,BUS         ;READ STATUS
07A9 3299     2124         JB1   IOTSK1        ;END IF INPUT BUFFER FULL
07AB 5299     2125         JB2   IOTSK1        ;END IF UPI BUSY
07AD A5       2126         CLR   F1
07AE 72B1     2127         JB3   IOTSKB
07B0 B5       2128         CPL   F1            ;F1=1 SAYS DATA IS COMING (VS COMMAND)
07B1 12D1     2129 IOTSKB: JB0   IOTSKD        ;JUMP TO IOTSKD IF OUTPUT BUFFER FULL
07B3 B6BE     2130         JF0   IOTSKC        ;JUMP IF ALREADY SENT COMM TO UPI
07B5 4330     2131         ORL   A,#030H       ;SEND BACK UPI STAT ORED WITH 030H
07B7 91       2132         MOVX  @R1,A         ;SEND COMMAND (R1 IS PHONEY ADDR)
07B8 95       2133         CPL   F0            ;SAYS COMMAND SENT
07B9 E499     2134         JMP   IOTSK1
07BB FC       2135 IOTSKC: MOV   A,R4          ; NO , WHAT IS COMMAND
07BC F2C5     2136         JB7   IOTSKE        ;IF THIS BIT SET, HAD IO ERROR
07BE D303     2137         XRL   A,#IORDB      ;SEE IF IT WAS IO READ (FROM BAM)
07C0 96DD     2138         JNZ   IOERRM        ;JUMP IF NOT (MUST BE)
07C2 9900     2139         ANL   P1,#000H      ;GOING TO SEND DATA TO UPI, CLR P10
07C4 FD       2140         MOV   A,R5          ;THIS IS BAM DATA
07C5 91       2141 IOTSKE: MOVX  @R1,A         ;ADDR IN R1 IS PHONEY, NEED A/R
              2142                               ;IF JUMP TO HERE, HAD ERROR, P10=1, A
              2143                               ;HAS ERROR COMMAND
07C6 B91F     2144 IOEND:  MOV   R1,#R7FORG    ;IF TEST BIT ON, STAY IN I/O
```

BAD ORIGINAL

```
LOC  OBJ         LINE        SOURCE STATEMENT

07C8 F1          2145            MOV    A,@R1
07C9 1296        2146            JB0    IOTSKZ          ;JUMP IF YES
07CB 74C4        2147            CALL   DSENP6          ;TURN OFF ENAIO
07CD 23FB        2148 IOEND2:    MOV    A,#NIOTSK       ;GET READY TO TURN OFF WAKEUP BIT
07CF E476        2149            JMP    MAINR
07D1 9900        2150 IOTSKD:    ANL    P1,#000H        ;OUTPUT BUFFER FULL, 1ST TURN OFF P10
07D3 08          2151            INS    A,BUS           ;READ COMMAND/DATA
07D4 55          2152            STRT   T               ;DONE WITH P6, RESTART T
07D5 76F0        2153            JF1    ITSK3J          ;JUMP IF DATA
07D7 AC          2154            MOV    R4,A            ;IT'S COMM, STORE IN R4
07D8 07          2155            DEC    A               ;CHECK TO SEE IF LEGAL COMMAND
07D9 53FC        2156            ANL    A,#0FCH
07DB C6DF        2157            JZ     ITSKD1          ;JUMP IF LEGAL
07DD 84CE        2158 IOERRM:    JMP    IOERR
07DF FC          2159 ITSKD1:    MOV    A,R4            ;IT'S LEGAL, SEE IF IORDB
07E0 D303        2160            XRL    A,#IORDB
07E2 9698        2161            JNZ    IOTSK8          ;JUMP TO GET DATA WORD IF NOT IORDB
07E4 74C4        2162            CALL   DSENP6          ;MUST DO THIS BEFORE READING RAM
07E6 343C        2163            CALL   REDRAB          ;IT'S IORDB, READ WORD
07E8 AD          2164            MOV    R5,A            ;STORE IN R5
07E9 B937        2165 IOTSK5:    MOV    R1,#IOPTR       ;GET READY TO INCR IOPTR
07EB 11          2166            INC    @R1             ;DO IT
07EC 76C6        2167            JF1    IOEND           ;JUMP HERE IF END OF IOWRB
07EE E495        2168            JMP    IOTSKZ
07F0 84AE        2169 ITSK3J:    JMP    IOTSK3
                 2170            END
```

USER SYMBOLS

```
AKBREG 0331   AKREG1 0338   AKREG2 034C   AKREG4 0351   AKREG5 0376   AKREG7 0378   AKREG9 0385   AKREGA 038C
AR2    03A7   AR3    03B4   ARADD  03BA   ARITLP 03A9   ARKBNC 03A0   ARRNC  03A2   ARTLP1 03BE   BAKREQ 03CE
BAYSTA 0002   BAREG  0010   BBACT  0040   BBATL  0030   BCHECK 002C   BCOUNT 0050   BDREG  0000   BFAULT 002
BKBREG 0650   BLKKB  0484   BMAXPD 002E   BMETYP 002F   BRGONE 00A3   BRGTHO 00A0   BTAREG 0090   BTDREG 007
BTDPPS 0078   BTEMP  0090   BTOTAL 0020   BTTDR  0080   CCE10  0253   CHAREN 0300   CHFAUL 05EC   CHGDIR 001
CHA7P9 0276   CHKTOT 04D8   CKBCHR 006A   CLR000 0163   CLR001 0170   CLRBAT 0205   CLRKYR 015E   CLRLSH 00E
CLRMSH 000F   CMPBA0 0316   CMPBA1 0319   CMPBA2 032F   CMPBA3 0327   CMPBA4 0325   CMPLOT 0310   COMPBA 031
COMPSM 030A   COMPEN 030E   CRBAT1 02D8   DISALP 0281   DISCHA 021B   DISFRG 00FC   DISKBR 048E   DISPAL 02E
DMPCLU 040E   DMPXIN 0001   DMPRIN 0000   DMPTAP 0005   DON333 0065   DONE   0394   DONE11 005D   DONE3  008
DONE31 0096   DONE32 008F   DONE33 00AE   DONE34 0088   DONE35 00C0   DONE36 0002   DONE37 00AC   DONE38 00D
DONE39 00D8   DONE3D 00A3   DONE3E 00A5   DONE3F 00CA   DONE5  0400   DONE51 040C   DONE52 0412   DONE53 041
DPROG  003A   DSALP1 022E   DSENP6 03C4   DSFRG1 0105   DSKB01 0401   DSKBRB 049A   DSKBRF 040D   DSKBRS 040
EBAM1  002F   EBAM12 000F   EBAM2  001F   ENAFRG 0305   ENAIO  0009   FAULT  0124   FAULT0 03E1   FAULT1 03E
FAULTC 03DD   FDBCK  0424   FDBCK1 0421   FDBK1  042D   FDBK2  0428   FINBAT 0292   FINNO  0267   FINNO1 027
FINNO2 0255   FNCLK5 02E4   FNDCHG 039C   HFAULT 05E4   INHKBI 0001   INIT   05E1   INIT1  05E7   INIT4  076
INITF  0774   INITP0 074C   INITP2 075C   INITPF 0761   INITPL 0737   INITPX 073E   INITZE 0000   IOABK2 04C
IOAKP  0002   IOAKB1 04D6   IOEND  07C6   IOEND2 07CD   IOERR  04CE   IOERRM 07DD   IOPTR  0037   IORDB  000
IOSPTR 0001   IOETSK 0794   IOSTA  0016   IOTSK  0004   IOTSK0 0798   IOTSK1 0799   IOTSK3 04AE   IOTSK5 07E
IOTSK6 04BE   IOTSKB 07B1   IOTSKC 07B8   IOTSKD 07D1   IOTSKE 07C5   IOTSKS 07A6   IOTSKZ 0796   IOWRB  03F
IOWRB1 03F5   IOWRB2 03FD   IOWRBJ 04D2   IOWRBX 0004   ITSK1J 04D0   ITSK3J 07F0   ITSKD1 07DF   KBTLG  01E
KBTLG2 01C3   KBTL04 01C1   KEY000 0212   KEY001 0216   KEY002 021A   KEYBD  0204   KEYBDI 0790   KEYDF  02
KEYSTA 0034   KEYTSK 0008   LSTC07 002B   LSTC09 002D   LSTCSM 002F   MAIN   077F   MAIN2  078A   MAINJ2 02
MAINR  0776   MAINR2 0779   METYPE 0017   MINONE 00FF   MOVRB1 0175   MOVRB0 017E   MOVRBS 0178   MPST1  00
MREB1  0188   MREB2  018C   MREB3  018F   MREB4  0198   MREB5  019C   MREBL  0192   MVIDDR 0633   NDONEP FF
NIOTSK FFFB   NKBIOT FFF3   NKYIOP FFF2   NODISP 00DE   NORINT 01ED   NPETHC FFB0   NPETTI FF9F   NPHOCL FF
NPTAEN FF9F   NPTTSK FFDF   NSTEP  0674   NSTEP0 067E   NSTEP1 067F   NSTEP2 068D   NSTEP3 068F   NSTEP5 06
NSTEP6 0692   NSTEP7 0699   NSTEP8 0690   NSTEP9 0695   NSTEPE 068E   NSTEPI 0675   NSTPM1 FF7F   NSWTSK FF
```

0019515


```
ONESHT 000F  PGMOVD 03CE  PFTSK  0680  PHOCL  0002  PORG01 0288  POB000 029C  POBAT1 0290  POBATL 0295
POSDON 0060  PRCTR  0036  PRINT  0517  PRINT1 05B7  PRINT3 0534  PRINT7 05C9  PRINT8 05C8  PRINT9 05CF
PROG1  0020  PROG2  0010  PROG2F 001F  PRTFA0 0554  PRTFE3 0556  PRTFA4 0564  PRTFE5 0558  PRTFA6 0568
PRTFA7 0559  PRTFA8 055B  PRTFAA 055E  PRTFXX 0565  PRTFXY 0570  PRTSTA 0035  PTAEN  0640  PTITSK 0020
PTK200 06CD  PTK201 06D3  PTK202 06D4  PTK204 0623  PTK20D 06DA  PTK211 0592  PTSK2  053A  PTSK05 0545
PTSK1  054B  PTSK2  0574  PTSK25 0625  PTSK26 0620  PTSK26 0580  PTSK25 0583  PTSK21 052C  PTSK2P 0689
PTSK32 059B  PTSK33 05A0  PTSK38 05A2  PTSK39 05AE  PTSK3A 05B1  PTSK3X 059E  PTSK41 05F8  PTSK42 05FE
PTSK4E 05D1  PTSK4F 05DA  PTSK4H 05D6  PTSK4P 05F9  PTSK4T 05EB  PTSK51 061F  PTSK55 062D  PTSK57 062A
P2FOPS 001P  P7FOPS 001F  RAP001 0145  RDEM51 015E  RDG001 0489  REAC01 0141  REDPM1 013F  REDBME 013C
REDPMS 0150  REG07  002A  REG09  002C  REGDSJ 0792  REGDSP 0434  REGREG 03CC  REGSH  002E  RGD003 045D
RGD005 0465  RGD09E 046F  RGD00A 043D  RGD00D 044F  RGD0BE 044D  RGDSP1 0457  RGDSP3 0455  RGDSP4 0475
RGDSPP 0479  SAVACC 0029  SERIPH 0092  SND    01D1  SND6   01DF  SND1   01E2  SND2   01E5  SND3   01EA
SOLEN  0042  SOLNFF 064F  STEPIN 0638  STEPTK 003C  STINTV 0004  STPDON 0020  STPM1  0080  STPTK2 003E
STPTRY 0042  SWRIBA 067F  SWTSK  0010  TIM000 0054  TIMCNT 00FB  TIMFIN 00EA  TIMIN2 003E  TIMIN1 0048
TIMINE 0059  TIMINT 0067  TIMPET 00FE  TMINT1 001B  TMINT2 0039  TMINT3 0026  TOFESS 01EB  TOFEST 01EA
TOPTI  01A0  TOPTIC 01AC  VER    0639  VER1   064A  VER2   0653  VER3   066A  VER5   065C  VER8   066F
VERLP  0645  WAIT   06FB  WAIT1  072F  WR8000 011D  WR8001 0126  WRITB1 0112  WRITB2 0133  WRITB5 0139
WRITBA 000F  WPTEM1 0102  WRTEMP 0105  WRTDSP 00EF  WTDSP1 00E8  XCHAPT 0565  ZEROSC 0003
```

ASSEMBLY COMPLETE.   NO ERRORS

BAD ORIGINAL

0019515

```
AMPEG  917    993#
AMPEG1 999#   1006
AMPEG2 1009   1011#
AMPEG4 1017#  1052   1056   1061   1758
AMPEG5 1028   1038#
AMPEG7 1016   1045#
AMPEG9 1049   1050   1053#  1060
AMPEGA 1047   1057#
AR2    1116   1118#
AR3    1124   1126#
ARADD  1122   1128   1130#
ARITLP 1119#  1135
ARYRNC 865    871    1023   1026   1113#  1729
ARRNC  878    1033   1114#  1451   1733   1736
ARTLP1 1134#
BAMPED 358    364    372    1157#  1236   1525
BAYSTA 71#    200    702    1157
BAREG  104#   1442   1721
BBACT  142#   898    1722
BBATL  111#   895    1728   1737   1738
BCHECK 105#
BCOUNT 113#   1735
BMREG  103#   1434   1454   1767
BFAULT 106#   1189   1976
BKEREG 114#   547    896    1000   1113   1340   1363   1898
BLKKB  1287   1336   1342#
BMAXPO 109#
BMETYP 110#
BRDONE 124#   1734
BRSTHD 125#   1730   2005
BTAREG 119#   976    975    1022   1029   1444   1720
BTDREG 116#   864    877    1025   1031   1436   1448   1766
BTDRP2 117#   539
BTEMP  121#   970    872    873    880    882    885    1027   1030   1034   1440   1446   1449   1452
BTOTAL 105#   886    1438
BTTMP  126#   1455
CCE10  799#   903    905
CHAPEN 921#
CHFAUL 1661   1782#
CHGDIP 39#
CH789  754    818#
CHKTOT 561    1021   1434#
CHRCHR 445#   992
CL0000 550#   551
CL0001 557    559#
CLPEAT 953    992#
CLPKYR 547#   645    854    900    1959
CLPLSH 36#    608    611    967    1844   1872
CLRMSN 75#    219    227    418    453    494    537    887    1059   1318   1807   1817   1848   1903   1986
CMPBA0 944    951#
CMPBA1 949    954#   964
CMPBA2 955    959#
CMPBA3 959    964#
CMPBA4 961    963#
CMPLOT 946#   2007
COMPBA 950#   1435   1439   1443
COMPGM 883    942#   1035   1453
COMPRH 945#   1981   1993
CREAT1 894#   1411
DISALP 826#   828    836
DISCHA 739    741    751#
DISFRG 434#   697    731    860    1283   1499   2016   2108
DISKER 984    1036   1363#
```

BAD ORIGINAL

0019515

```
DISPAL  742    910#
DMPCLU  131#
DMPKIN  133#   277
DMPPIN  132#
DMPTAP  134#
DON333  342    344#
DONE    267    313   1092#  1227
DONE11  258    263#
DONE2   260    297#
DONE31  302#   305
DONE32  309#   310
DONE33  338#   347
DONE34  345    348#
DONE35  353#   365
DONE36  351    363#
DONE37  331#   373
DONE38  371#
DONE39  329    330    366#
DONE3D  322    324#
DONE3E  320    321    323    325#
DONE3F  356    359#
DONE5   290   1223#
DONE51 1229#  1237   1239
DONE52 1228   1232#
DONE53 1234   1238#
DPROG    19#  2121
DSALP1  832    837#
DSEHPS  436   1145#  1397   1705   2147   2162
DSFRG1  440#  1173
DSKBP1 1376   1379#
DSKBP2 1370   1372#
DSKBPF 1381   1387#
DSKBPS 1383   1385#
EBAM1    16#   477    530
EBAM12   17#   517
EBAM2    18#
ENAFPG  852   1169#  2011   2056
ENAIO   155#  2119
FAULT   484#   522    535
FAULT0  497    563   1183#  1708
FAULT1  485   1181   1185   1188   1191#  1624
FAULTC 1180#  1514
FDBCK   254    279    305   1258#
FDBCK1  329   1224   1225   1226   1254#
FDBK1   311    312   1263   1266#
FDBK2  1259   1261#
FINEAT  775    848#
FINNO   758    760    763    765    767    769    771    773    778    805#   820    821
FINNO1  811#  1416
FINNO8  653    790#   839
FNCLK5  780    783    888    900#   915   1421   1567   1751   2021   2068
FNOCHG 1098   1100#
HFAULT 1703   1707#
INHKBI   83#    98   1524
INIT    163   1948#
INIT1  1952#  1953
INIT4  2050#  2052
INITF  2055   2058#
INITP0 2030#  2042
INITP2 2034   2035   2041#  2043
INITPF 2032   2044#
INITPW 1625   2018#
INITPX 2020   2023#
```

BAD ORIGINAL

INITZE	183#							
IOABX2	1410	1412#						
IOAVE	153#							
IOAP1	1400	1421#						
IOEND	2144#	2167						
IOEND2	1405	2148#						
IOERR	1204	1413	1417#	2158				
IOERRM	2130	2158#						
IOPTR	68#	1402	2114	2165				
IOPTP	154#	2137	2160					
IORFTF	152#							
IORTSY	2070	2100#						
IOSTF	67#							
IOTSK	88#	89	363					
IOTSKP	1410	2111#	2161					
IOTSK1	2113#	2124	2125	2134				
IOTSK3	1396#	2169						
IOTSK5	1210	2165#						
IOTSK6	1401	1405#						
IOTSKB	2127	2129#						
IOTSKC	2130	2135#						
IOTSKD	2129	2150#						
IOTSKE	2136	2141#						
IOTSKS	2122#							
IOTSKZ	2109#	2146	2168					
IOWRE	1202#	1420						
IOWRE1	1203	1205#						
IOWRE2	1207	1210#						
IOWREJ	1400	1419#						
IOWRSY	155#							
ITSK1J	1390	1418#						
ITSK3J	2153	2169#						
ITSKD1	2157	2159#						
KBTLG	639#	1037						
KBTLG3	644	646#	869					
KBTLG4	642	645#						
KEY000	738	740#						
KEY001	740	742#						
KEY002	742	744#						
KEYBD	731#	2083						
KEYBDJ	2079	2083#						
KEYDF	645	788	792#	889	911	912	914	1038
KEYSTF	53#	793	1755	1889	1995	2057		
KEYTSY	82#	96	99	371				
LSTC07	50#							
LSTC09	51#	781						
LSTCSW	52#							
MAIN	2074#	2081						
MAIN2	2077	2080#						
MAINJ2	797	799#	885	893				
MAINP	800	1335	1569	1686	2063#	2149		
MAINP2	798	1558	1672	2066#				
METYPE	55#	640	1045	1989	2053			
MINONE	38#	247	1145	1960				
MOVRB1	578#	881	897	899	1029	1447		
MOVRBE	584#	887	1437	1441	1445	1722	1768	
MOVRBS	580#	874	1456	1739				
MPST1	31#	390						
MREB1	579	586#						
MREB2	591	594#						
MREB3	593	595#						
MREB4	599	602#						
MREB5	601	604#						

BAD ORIGINAL

0019515

```
NESE[illegible]  59?    59?#   604
N?TD[illegible]  884    1719   1766#
NDONEP  144#   1811   1824   1853
NIOTSK   89#     98     99   2148
NKBIOT   99#
NKYIOP   98#    799
NDOISP  212    389#
NOPINT  187    696#
NPETHC   97#
NPETT1   98#    626
NPHOCL   94#     97   1663
NPTAEN   96#     97     96   1685
NPTTSK   87#     98
NSTEP   1641   1839#
NSTEP0 1846#   1874
NSTEP1 1847#   1859   1876
NSTEP2 1856#   1865
NSTEP3 1854    1862#
NSTEP5 1862    1866#
NSTEP6 1864#   1867
NSTEP7 1868#   1871
NSTEP8 1841    1878#
NSTEP9 1843    1875#
NSTEPF 1851    1857#
NSTEPI 1849#   2048
NSTPM1  141#   1864
NSWTSK   85#   1334
ONESHT  398#   1231
PGMOVD 1145#   2128
PFTSK    92#    782
PHOCL    93#     94    357
PORDX1  859     870#
POSB29  856     858#
PORTT1  855     857    859#
POBRTL  854#    859
POSDON  143#    144   1821
PRCTR    59#    194    622   1959   2089
PRINT  1495#   2082
PRINT1 1587    1668#
PRINT3 1511    1515#
PRINT7 1672#
PRINT8 1671    1673#
PRINT9 1673    1675#
PROG1    26#    667    788
PROG2    23#     25    207   1147
PROG2E   25#    473
PRTFA0 1541    1542   1551#  1589   1593   1594   1612   1613   1643
PRTFA3 1552    1559#
PRTFA4 1559    1561#
PRTFA5 1554#   1568
PRTFA6 1557    1563#
PRTFA7 1555#   1562
PRTFA8 1556#   1925
PRTFAA 1558#   1564   1696   1716
PRTFXX 1565#
PRTFXY 1503    1568#
PRTSTA   62#    658    795   1588   1555   1922   2025
PTAEN    95#     96   1235
PTTTSK   86#     87    282
PTK200 1909    1912#
PTK201 1911    1917#
PTK202 1913    1918#
PTK204 1891    1893#
```

BAD ORIGINAL

PTK20	1647	1748	1922#								
PTK2L1	1614#										
PTSK0	1523#										
PTSK05	1533#										
PTSK1	1518	1538#									
PTSK2	1517	1579#									
PTSK25	1894#	1897									
PTSK26	1901#	1921									
PTSK26	1584	1590#									
PTSK29	1592#	1640									
PTSK2L	1610#	1615									
PTSK2P	1618	1886#									
PTSK32	1586	1622#									
PTSK33	1623	1625#									
PTSK38	1591	1635#									
PTSK39	1637	1638	1641#								
PTSK3A	1617	1644#									
PTSK3X	1496	1624#									
PTSK41	1668	1713#									
PTSK42	1701	1717#									
PTSK4E	1662	1682#									
PTSK4F	1684	1687#									
PTSK4H	1685#	1690									
PTSK4P	1706	1714#									
PTSK4T	1669	1695#									
PTSK51	1675	1745#									
PTSK55	1749	1750	1755#								
PTSK57	1747	1753#									
R2FORG	43#	435	1170								
R7FORG	44#	1284	1419	1504	2018	2144					
RAB0E1	518#										
RDBM51	525	537#									
RDG021	1345#	1347									
REREP1	513	516#									
REDBM1	417	514#	598	1004	1121	1894					
REDBM2	512#	954	1977	2163							
REDBMS	529#	643	867	994	1119						
REG07	45#	732	1230	1956							
REG09	46#	268	552	735	1992						
REGDSI	2076	2084#									
REGDSP	1283#	2884									
REGRED	901	1017	1156#	1238							
REGSU	49#	362	1298	1526	1692						
RGD003	1308	1310#									
RGD005	1321#	1327									
RGD009	1323	1329#	1330								
RGD00A	1286	1288#									
RGD00D	1299#	1302									
RGD00F	1298#	1329									
RGDSP1	1305#	1341									
RGDSP3	1299	1303#									
RGDSP4	1293	1334#									
RGDSPP	1296	1336#									
SAVACC	69#	399	1955								
SEPIPH	33#	1386									
SND	661#	1538	1642	2030							
SND0	671#										
SNEM	670	674#									
SND2	669	677#									
SND3	677	681#									
SOLEN	14#	15	206	472	516	529	666	786	787	1146	1948
SOLNPF	15#	475									
STEPIN	72#	1467	1893	2044	2058						

BAD ORIGINAL

0019515

```
STEPTY   75#   1459
STINTV   145#  621
STPDON   142#  144   1812
STPM1    139#  141   1855
STPTK2   78#   1686
STPTRY   147#  1587
SWPIBA   22#   489
SWTSK    84#   85    1156  2017
TIM000   265   267#
TIMCNT   37#   192   1961
TIMFIN   261   269   282   398#
TIMIN0   220   223   233#
TIMIN1   222   246#
TIMIN5   226   241   273#
TIMINT   190#
TIMPET   362   394   397   399#
TMINT1   196   199   204#
TMINT2   216   227#
TMINT3   211#  288
TOFFSS   649   701   706#  1191  1565  1674
TOFFST   707#
TOPT1    621#  661   1782
TOPTIC   622#  1646  1700  1714
VER      1579  1781#
VER1     1794  1796#
VER2     1797  1803#
VER3     1818  1820#
VER5     1805  1810#
VER6     1804  1824#
VERLP    1792# 1802
WAIT     1971# 1972
WAIT1    2013# 2015
WPB002   476   478#
WPB001   483   485#
WRITB1   465   472#
WRITB2   493   495#
WRITB5   495   498#
WRITBA   21#   488
WRTBM1   464#  550   560   592   600   997   1005  1011  1134  1205
WRTBME   466#  594   602   605   1190  1209
WRTDSP   415#  425   1310  1372
WTDSP1   422   424#
XCHAPT   1465# 1515  1551  1616  1901  1918
ZERO9C   34#   1322

CROSS REFERENCE COMPLETE
```

BAD ORIGINAL

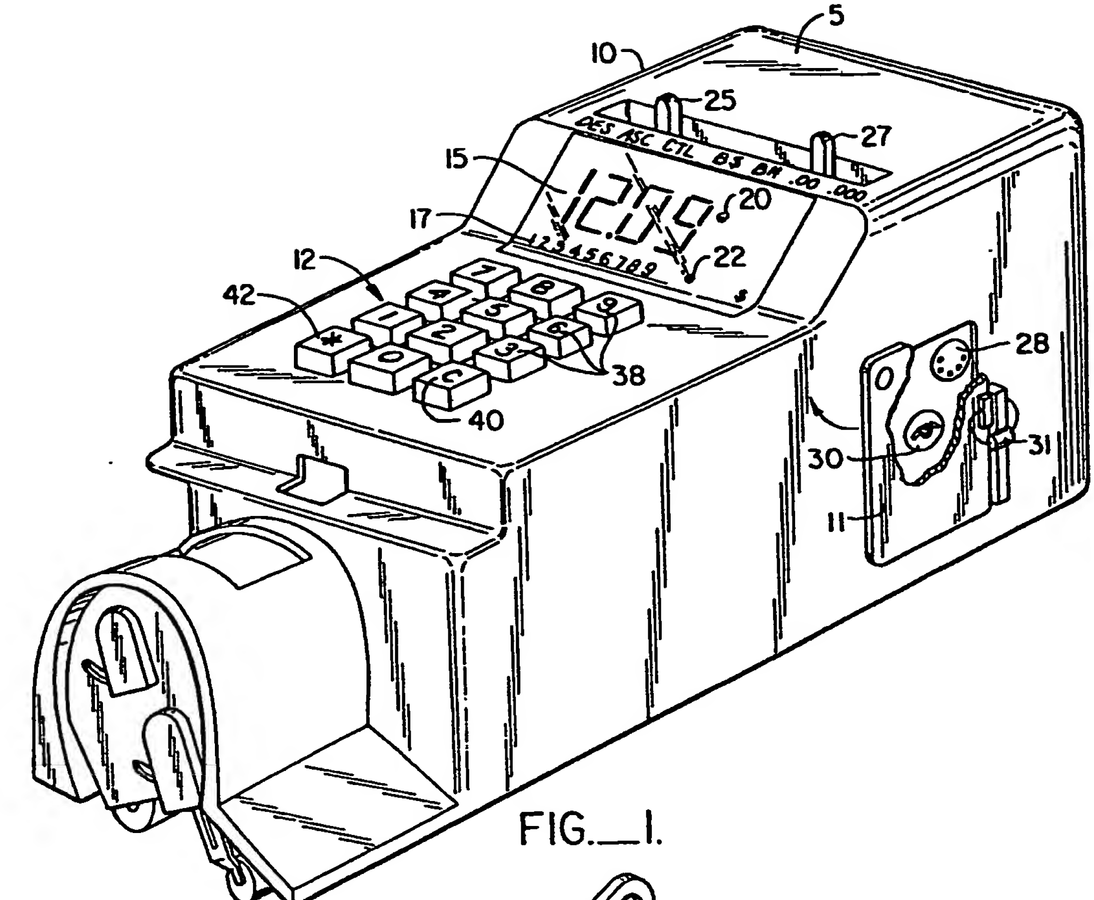

5
10
25
27
15
20
17
22
12
42
38
40
28
30
31
11
DES ASC CTL BS BM .00 .000
12345678 9

FIG._1.

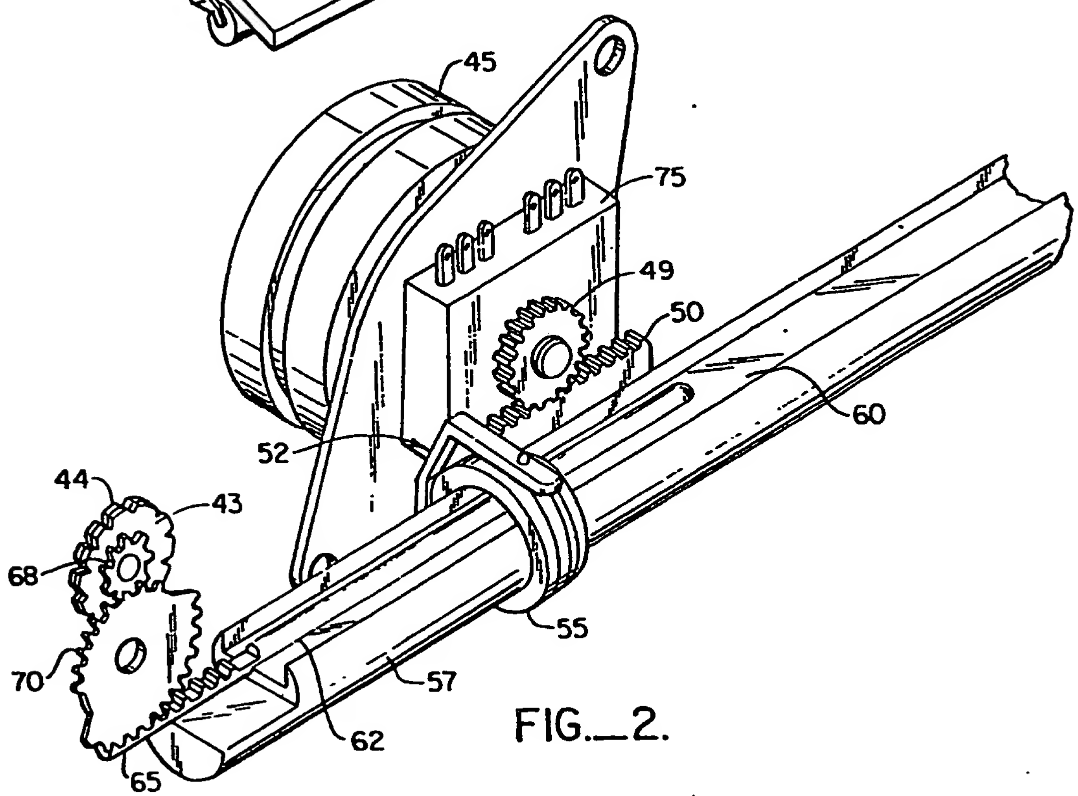

45
75
49
50
60
52
44
43
68
55
70
57
62
65

FIG._2.

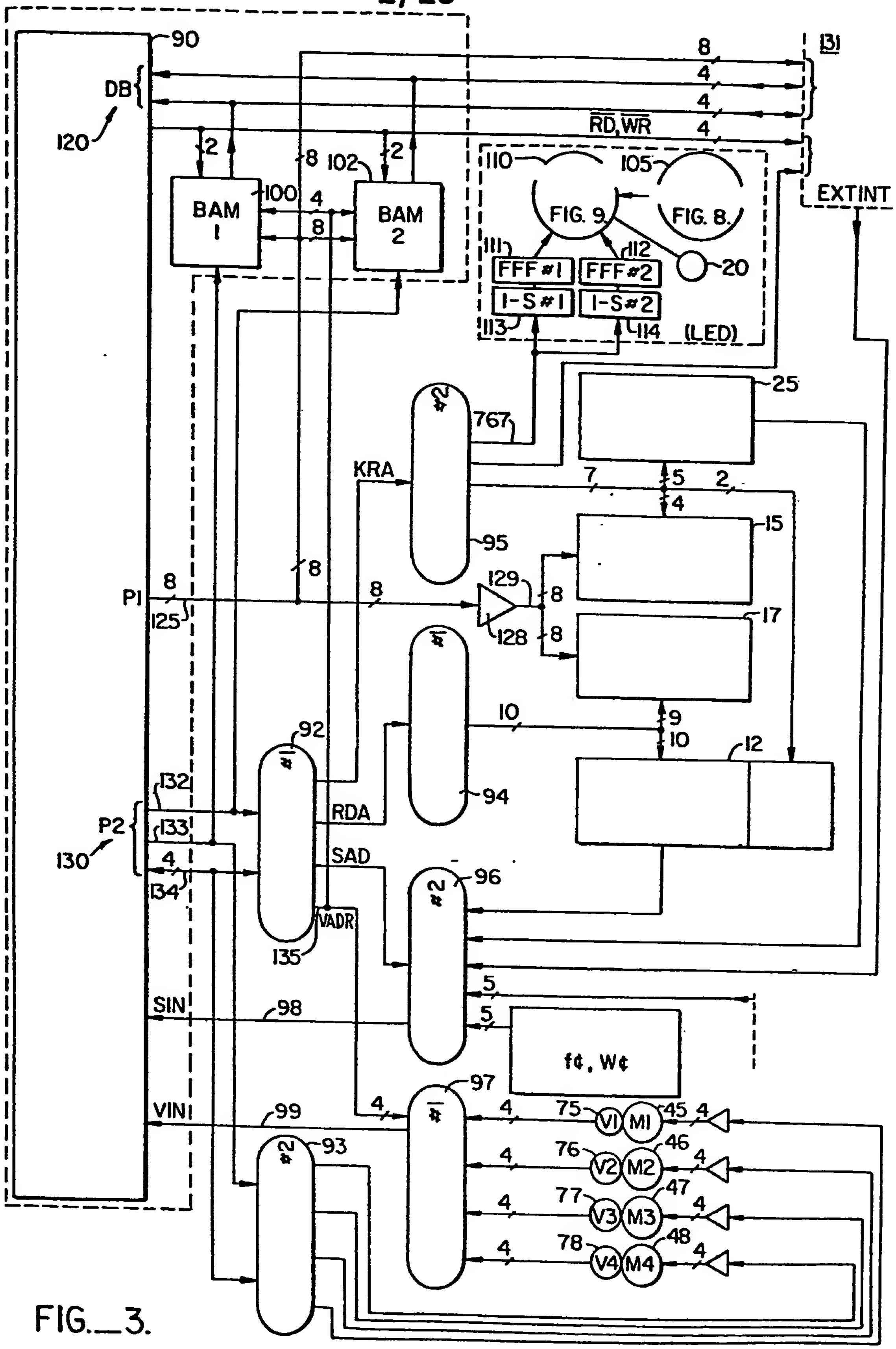

90
DB
120
BAM 1
100
BAM 2
102
RD, WR
131
EXTINT
110
105
FIG. 9
FIG. 8.
111
112
FFF #1
FFF #2
20
I-S #1
I-S #2
113
114
(LED)
25
767
KRA
15
95
P1
125
129
128
17
92
94
12
P2
130
132
133
134
RDA
SAD
VADR
135
96
SIN
98
f¢, W¢
97
VIN
99
93
V1
M1
V2
M2
V3
M3
V4
M4
75
76
77
78
45
46
47
48

FIG._3.

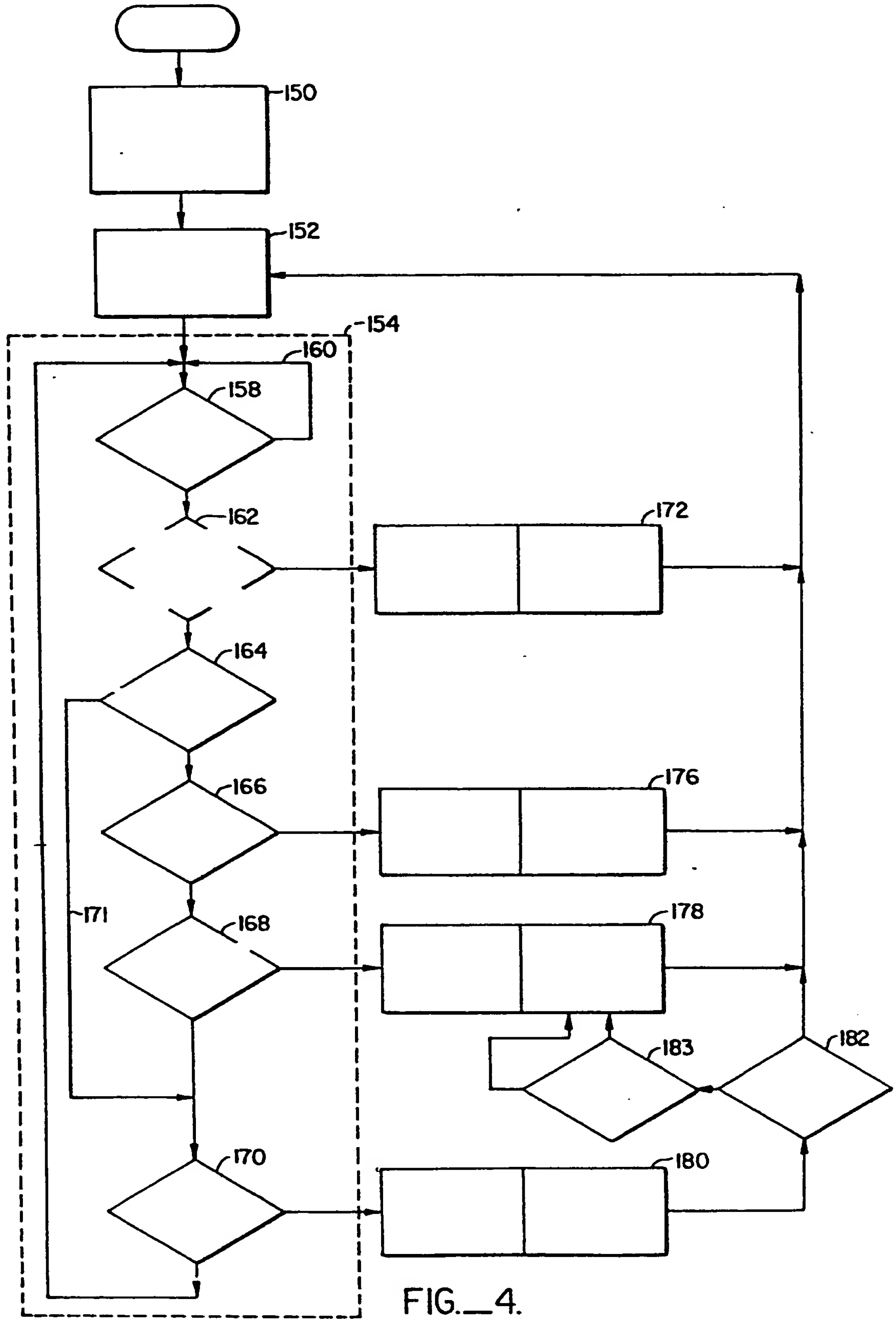

150
152
154
160
158
162
172
164
166
176
171
168
178
183
182
170
180

FIG._4.

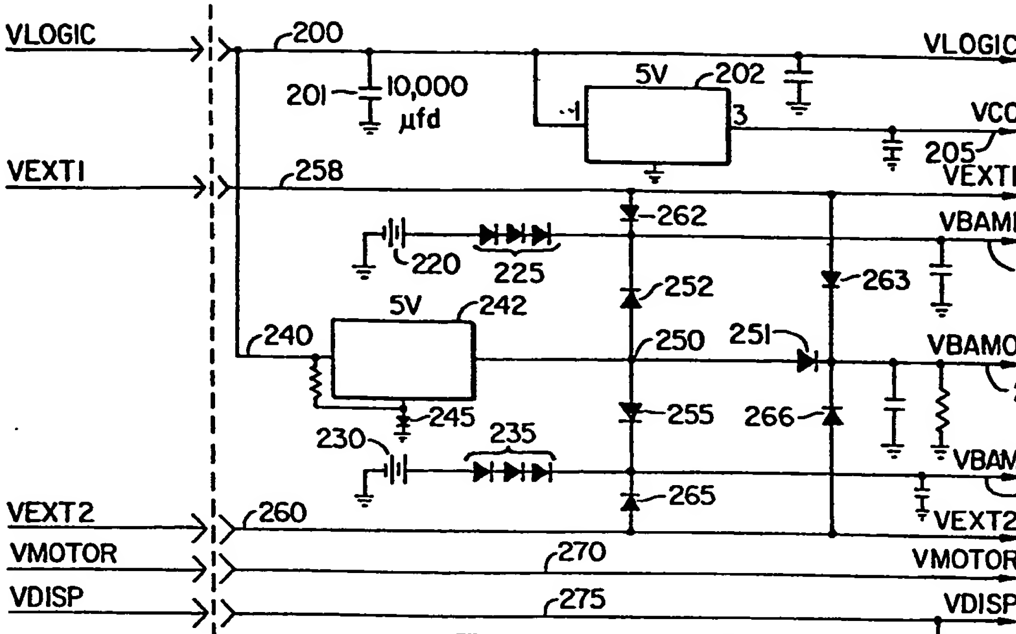


FIG._5.

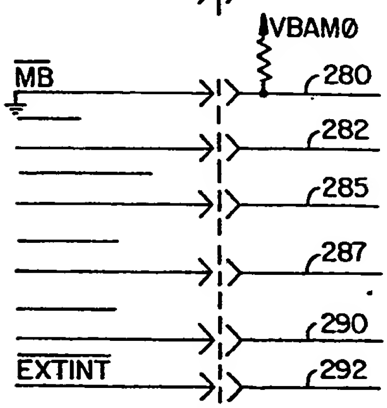


FIG._6.

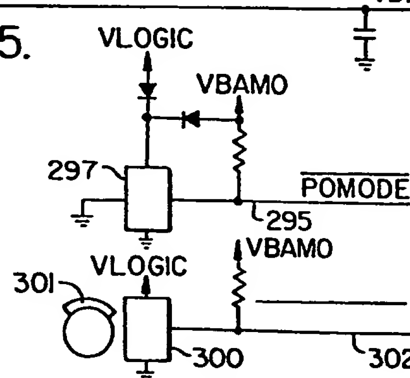


FIG._7.

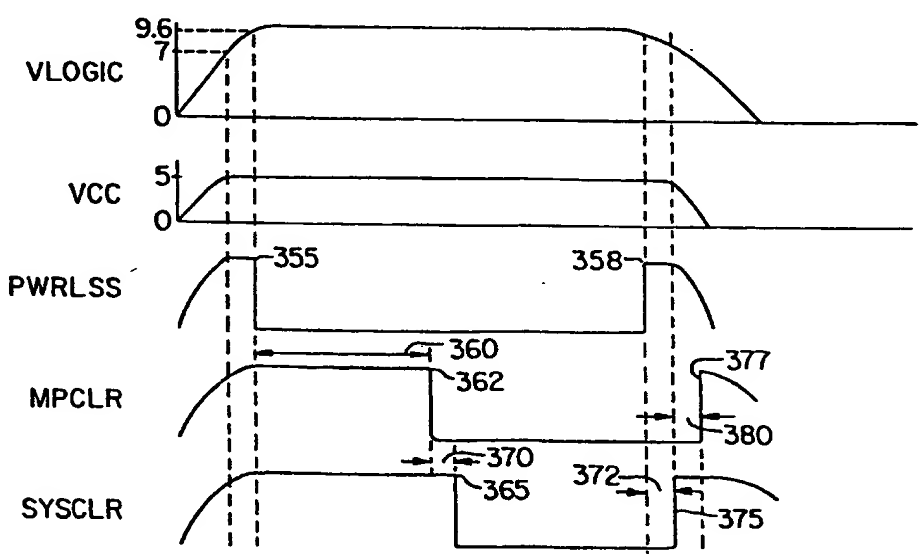


FIG._10.

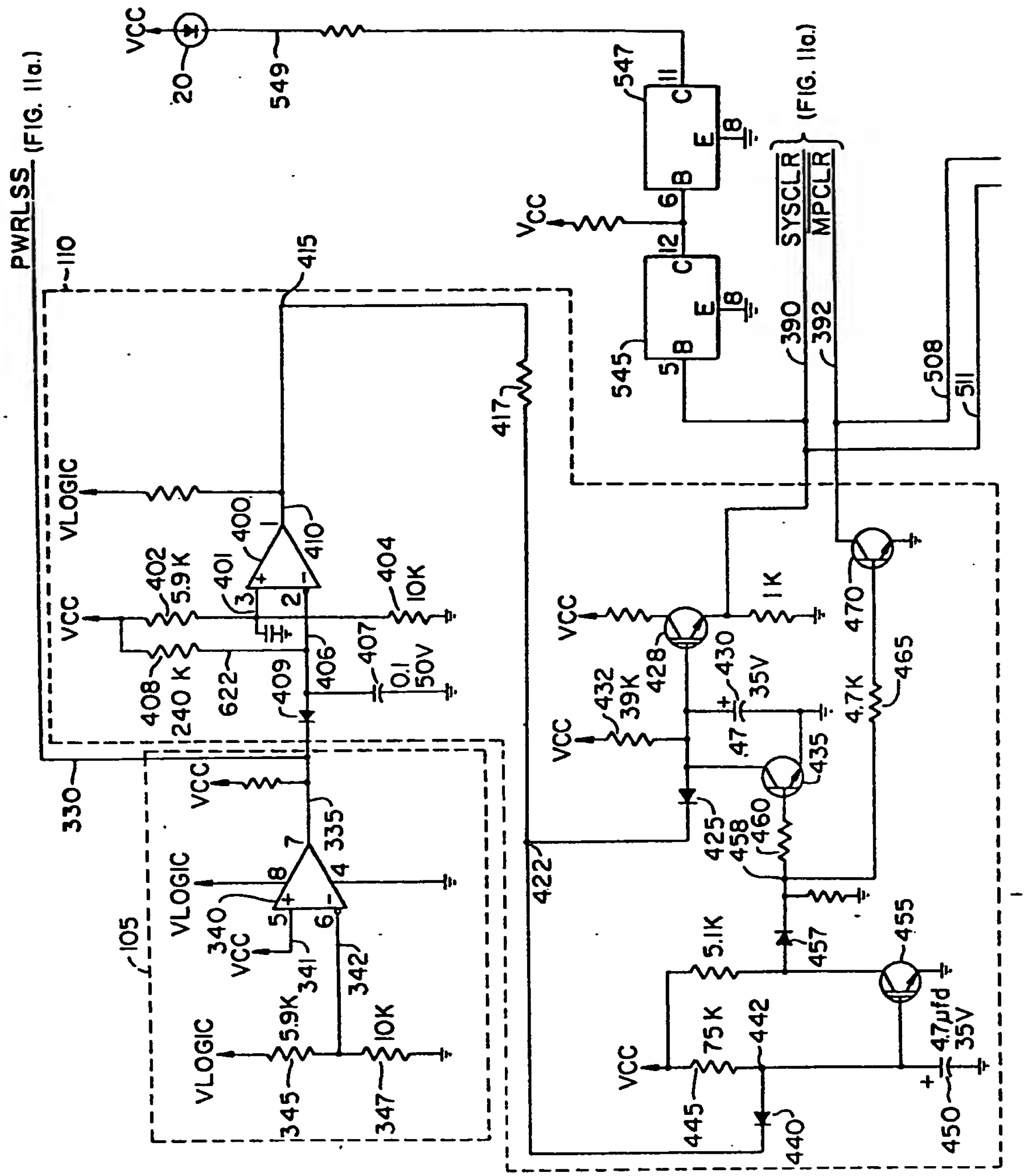

VCC
20
549
547
545
VCC
PWRLSS (FIG. 11a.)
110
415
VLOGIC
400
401
410
402
5.9K
VCC
404
10K
408
240 K
622
409
406
407
0.1
50V
417
330
105
VCC
335
VLOGIC
340
VCC
341
342
345
5.9K
347
10K
VLOGIC
SYSCLR
MPCLR
(FIG. 11a.)
390
392
508
511
VCC
428
430
35V
1K
470
465
4.7K
432
39K
47
435
425
458
460
422
5.1K
457
455
VCC
75 K
442
445
440
450
4.7μfd
35V

FIG. 8.

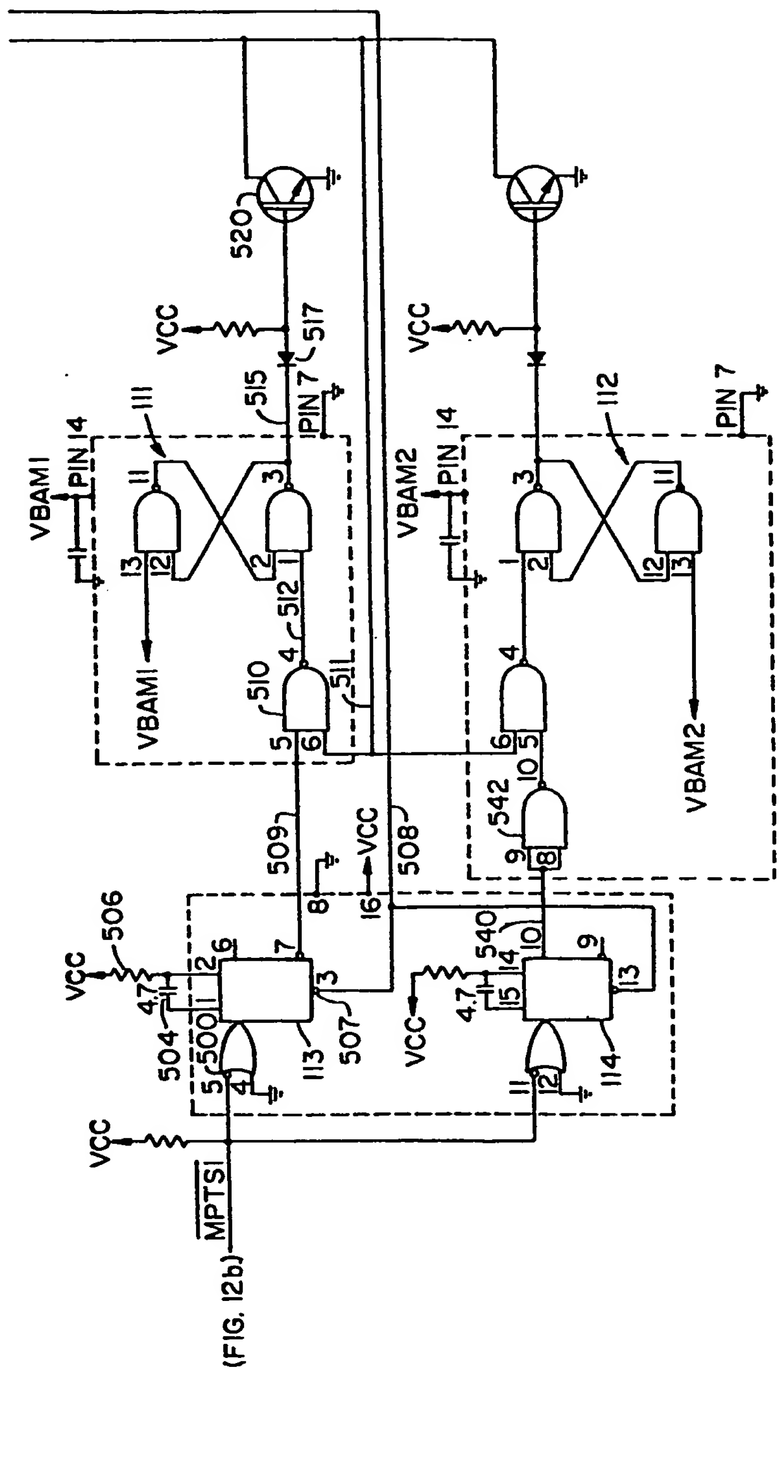


FIG. 9.

FIG. 8.
FIG. 9.

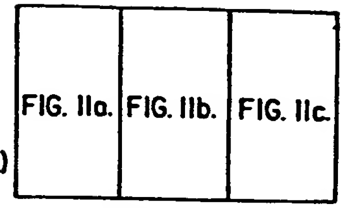


FIG._11.

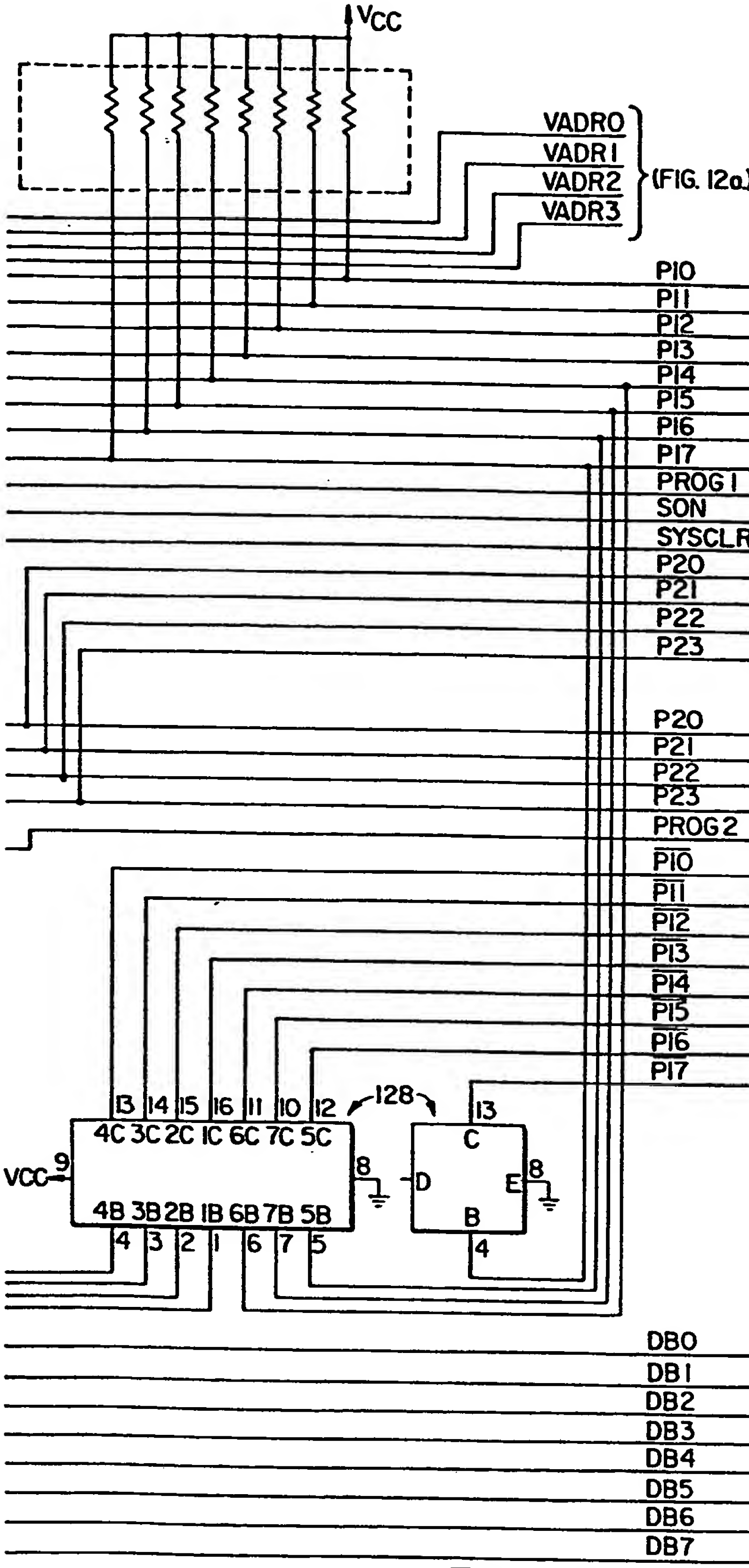
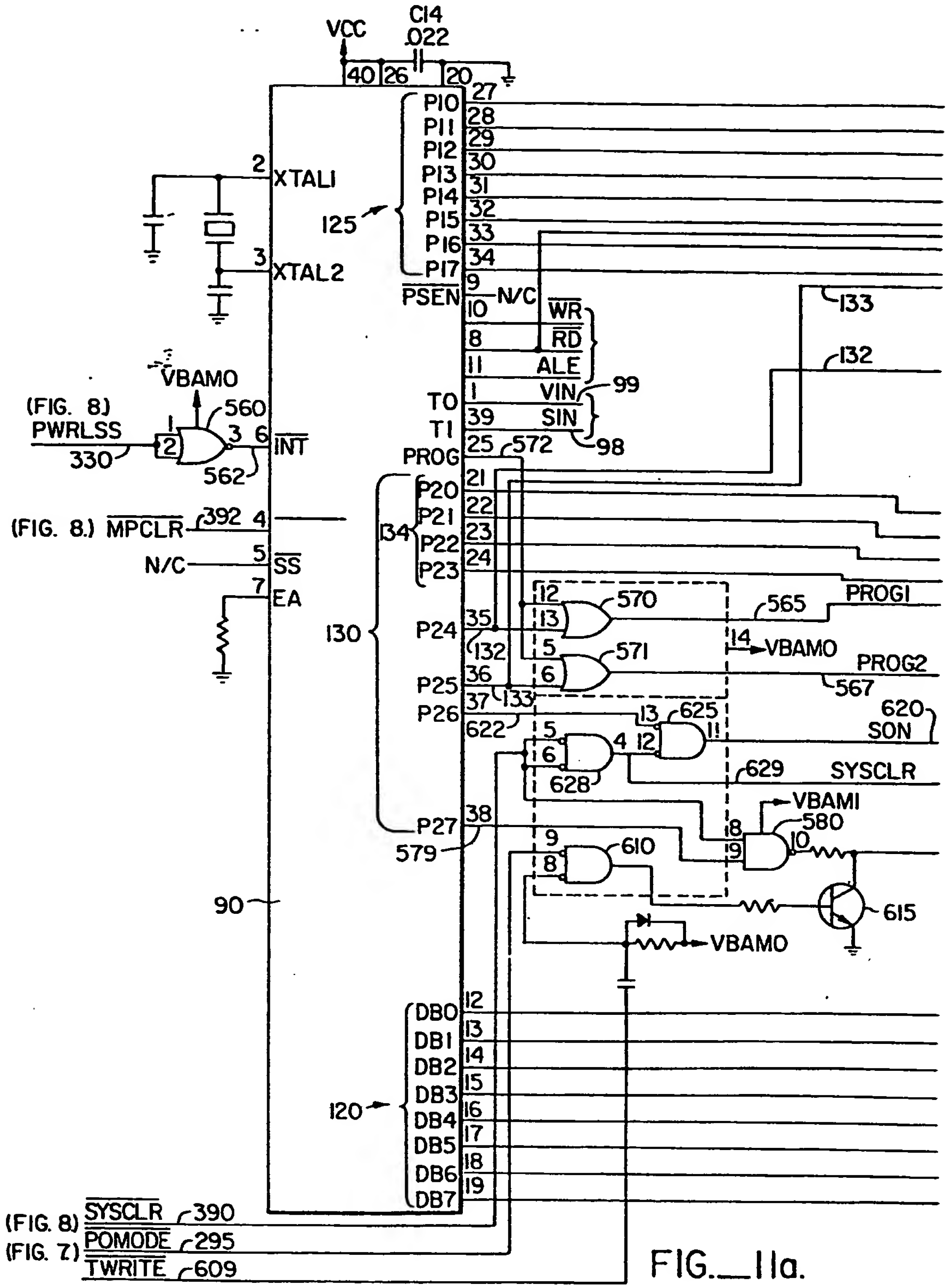


FIG._11c.



VCC
C14
.022
40
26
20
27
P10
28
P11
29
P12
30
P13
31
P14
32
P15
33
P16
34
P17
125
2
XTAL1
3
XTAL2
9
PSEN
10
N/C
WR
8
RD
11
ALE
1
VIN
99
T0
39
T1
SIN
98
VBAMO
(FIG. 8)
PWRLSS
330
1
2
3
6
560
562
INT
PROG
25
572
133
132
21
P20
22
P21
23
P22
24
P23
134
(FIG. 8.) MPCLR
392
4
N/C
5
SS
7
EA
130
P24
35
132
36
P25
37
133
P26
622
12
13
570
565
PROG1
5
6
571
14
VBAMO
PROG2
567
13
625
12
11
620
SON
5
6
4
628
629
SYSCLR
VBAMI
580
8
9
10
P27
38
579
9
8
610
615
VBAMO
90
DB0
12
DB1
13
DB2
14
DB3
15
DB4
16
DB5
17
DB6
18
DB7
19
120
(FIG. 8) SYSCLR
390
(FIG. 7) POMODE
295
TWRITE
609

FIG._11a.

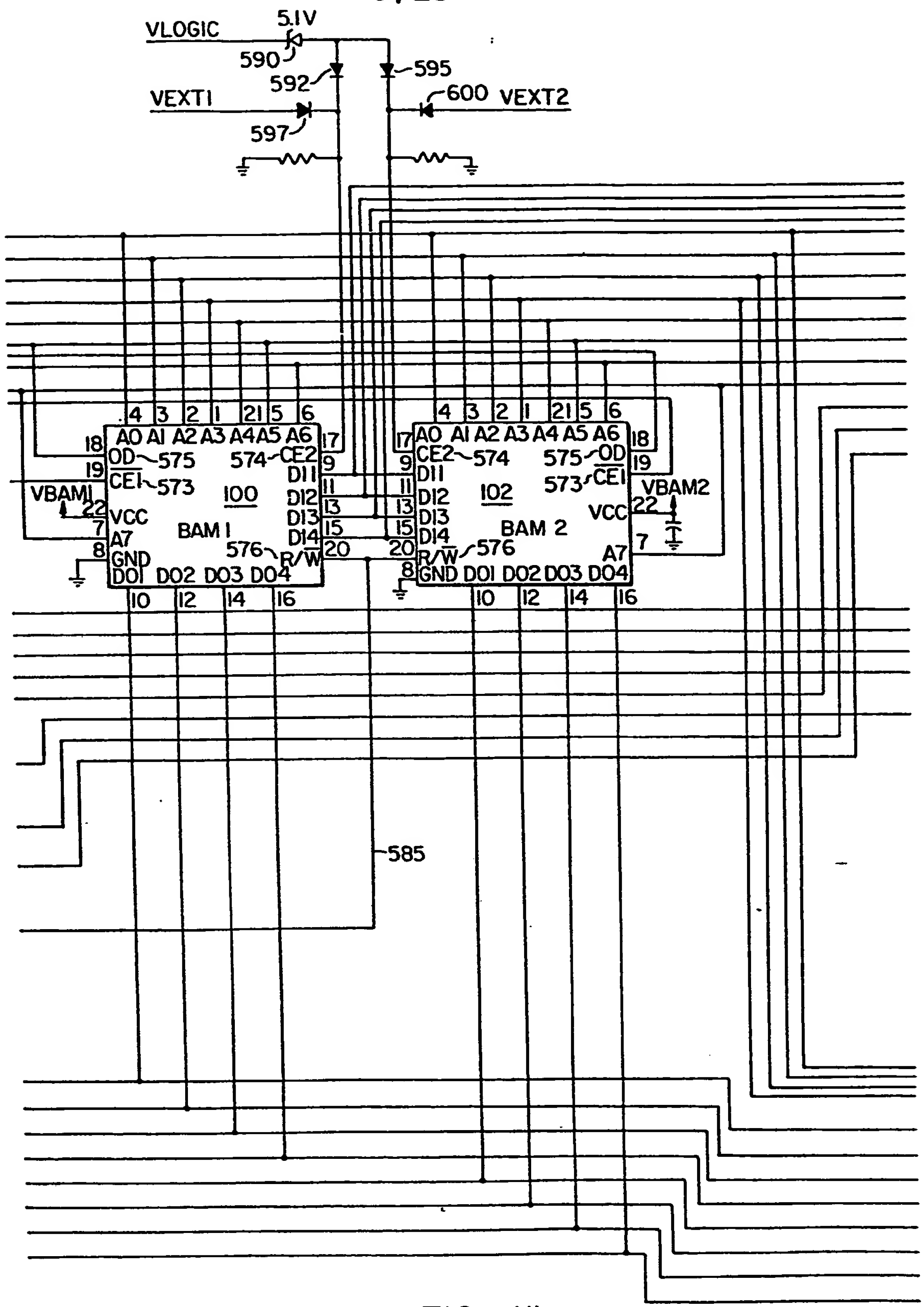
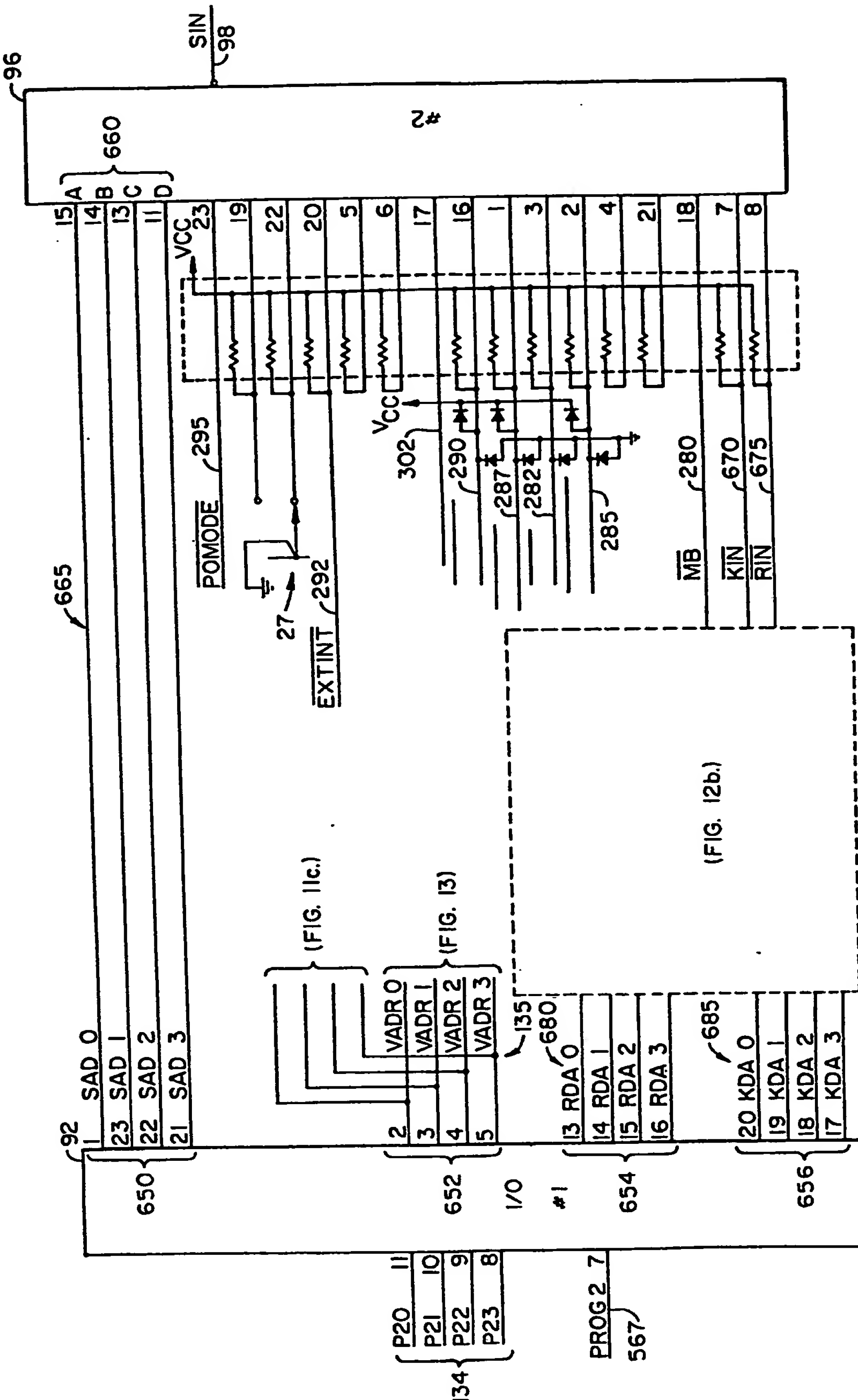

VLOGIC
5.1V
590
592
595
VEXT1
600 VEXT2
597
A0 A1 A2 A3 A4 A5 A6
OD 575
CE1 573
574 CE2
100
BAM 1
VBAM1
VCC
A7
GND
576 R/W
DI1 DI2 DI3 DI4
DO1 DO2 DO3 DO4
102
BAM 2
VBAM2
585

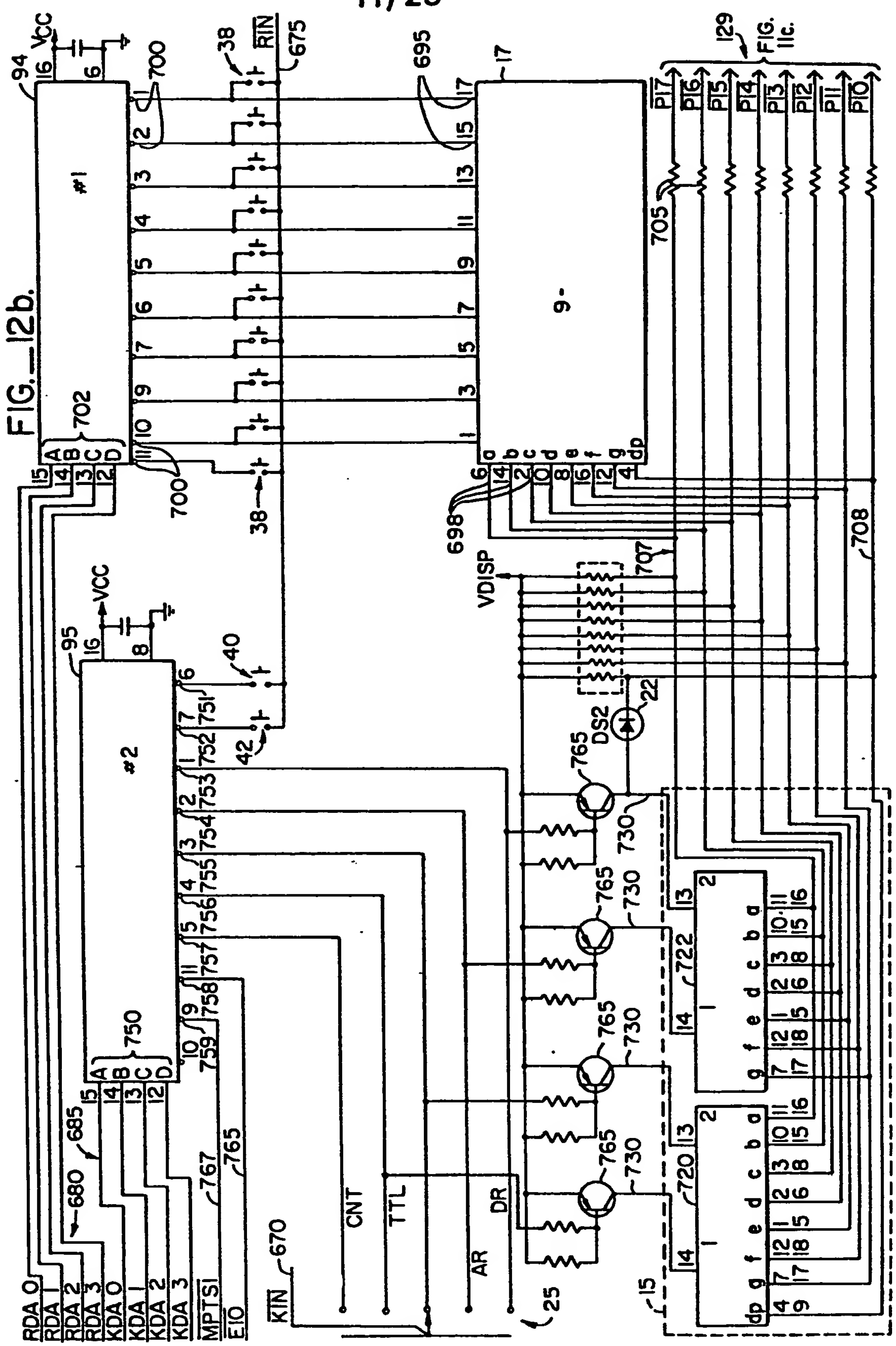
FIG._11b.



SIN
98
96
#2
660
A
B
C
D
15
14
13
11
VCC
23
19
22
20
5
6
17
16
1
3
2
4
21
18
7
8
665
POMODE
295
27
292
EXTINT
VCC
302
290
287
282
285
280
670
675
MB
KIN
RIN
(FIG. 12b.)
(FIG. 11c.)
(FIG. 13)
SAD 0
SAD 1
SAD 2
SAD 3
VADR 0
VADR 1
VADR 2
VADR 3
135
680
RDA 0
RDA 1
RDA 2
RDA 3
685
KDA 0
KDA 1
KDA 2
KDA 3
92
650
652
I/O
#1
654
656
P20
P21
P22
P23
134
PROG2
567

FIG._12a.

FIG._12b.



VCC
VDISP
RDA 0
RDA 1
RDA 2
RDA 3
KDA 0
KDA 1
KDA 2
KDA 3
MPTSI
EIO
KIN
CNT
TTL
AR
DR
RIN
DS2
FIG. 11c.

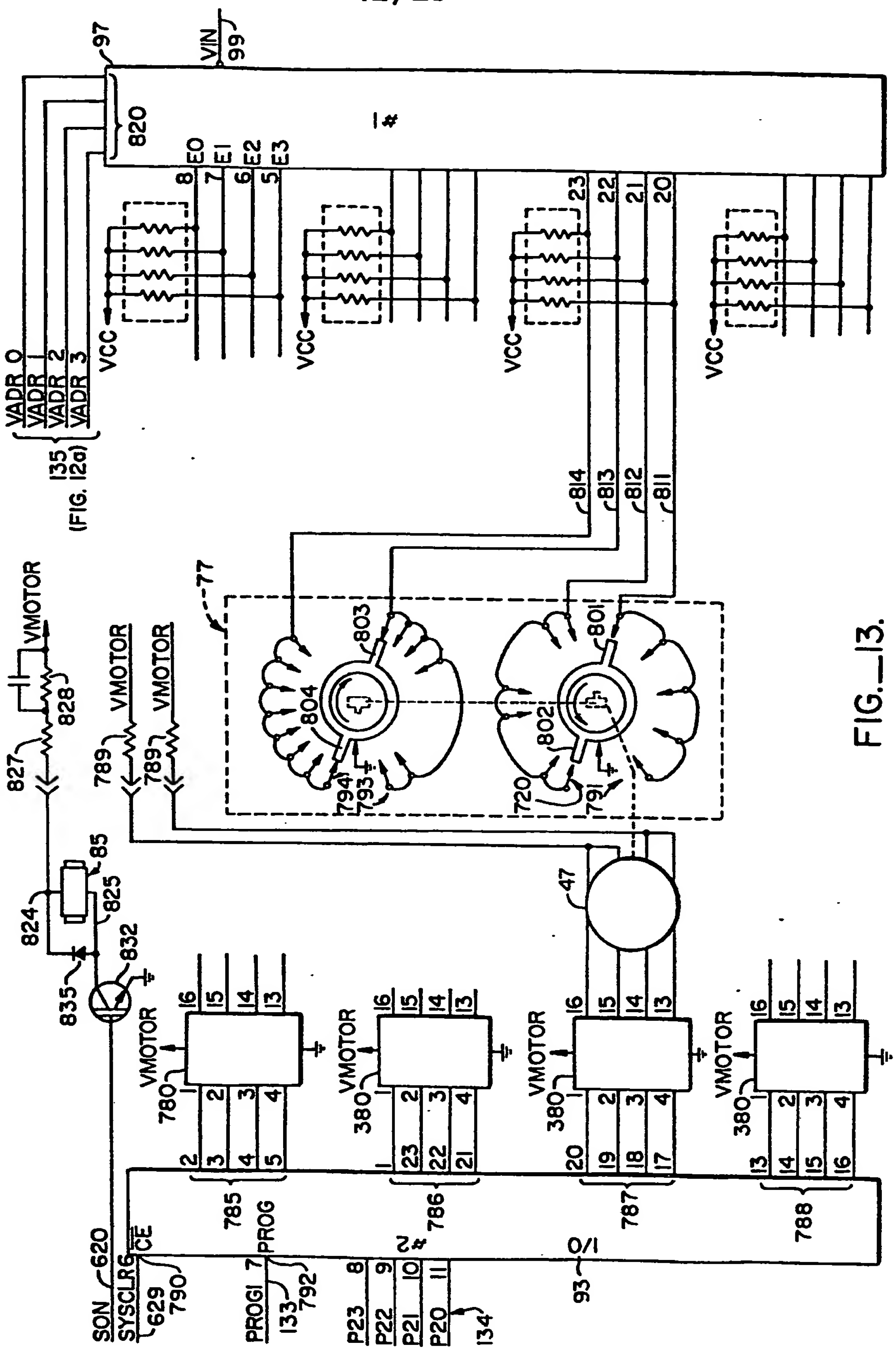
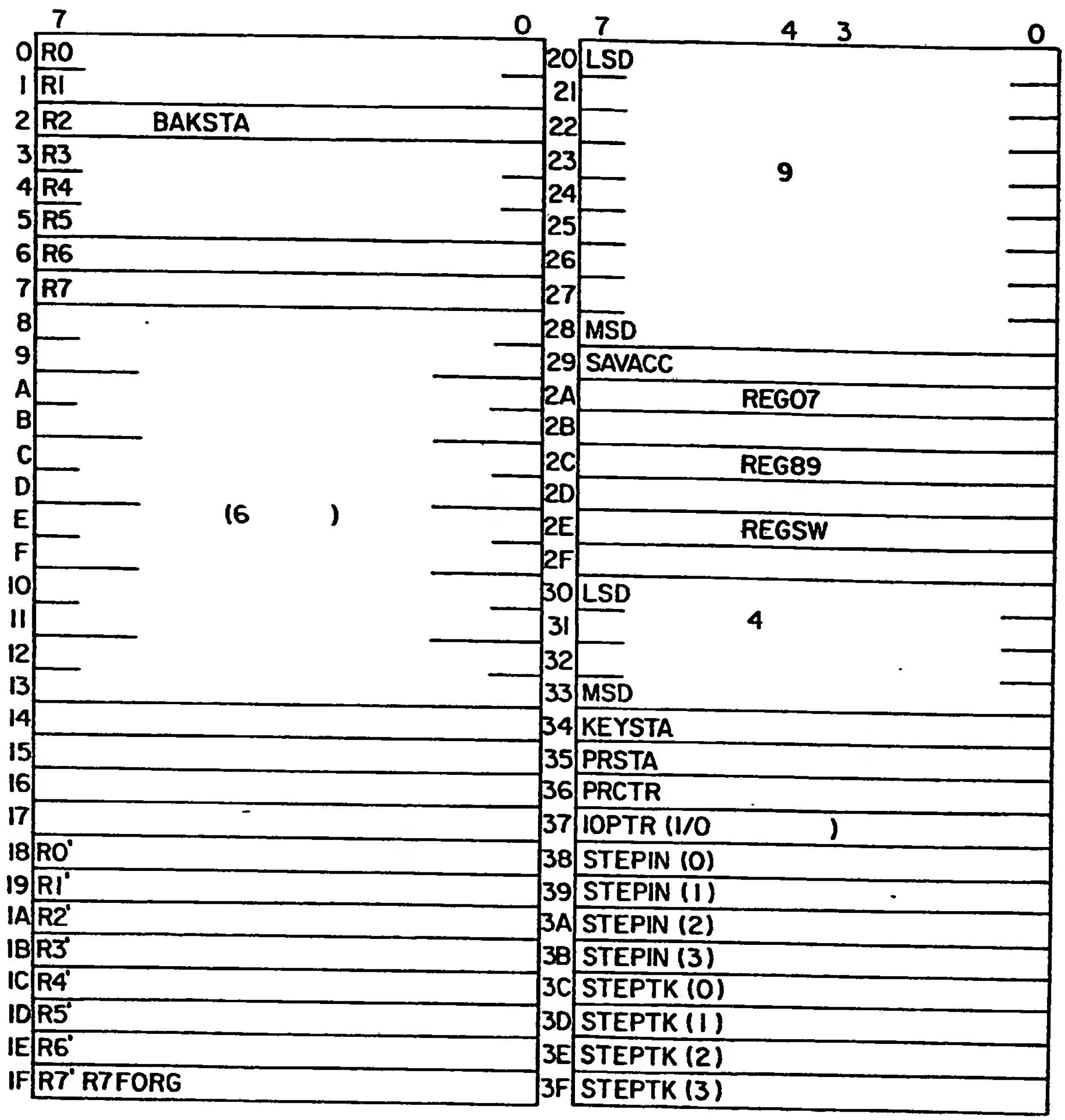

97
VIN
99
820
#1
E0
E1
E2
E3
VCC
VADR 0
VADR 1
VADR 2
VADR 3
135
(FIG. 12a)
814
813
812
811
77
803
804
801
802
794
793
720
791
VMOTOR
827
828
789
85
824
825
832
835
47
780
380
785
786
787
788
PROG
CE
#2
I/O
93
SON
SYSCLR6
620
629
790
PROGI
792
133
P23
P22
P21
P20
134

FIG._13.



7
0
0 R0
1 R1
2 R2 BAKSTA
3 R3
4 R4
5 R5
6 R6
7 R7
8
9
A
B
C
D
E (6)
F
10
11
12
13
14
15
16
17
18 R0'
19 R1'
1A R2'
1B R3'
1C R4'
1D R5'
1E R6'
1F R7' R7FORG
7
4
3
0
20 LSD
21
22
23 9
24
25
26
27
28 MSD
29 SAVACC
2A REG07
2B
2C REG89
2D
2E REGSW
2F
30 LSD
31 4
32
33 MSD
34 KEYSTA
35 PRSTA
36 PRCTR
37 IOPTR (I/O)
38 STEPIN (0)
39 STEPIN (1)
3A STEPIN (2)
3B STEPIN (3)
3C STEPTK (0)
3D STEPTK (1)
3E STEPTK (2)
3F STEPTK (3)

FIG._14.

BIT	R7FORG (IF HEX)	REG07 (2A HEX)	REG89 (2C HEX)	REGSW (2E HEX)
0		"0"	"8"	DR
1	EXTINT	"1"	"9"	AR
2	.000	"2"		
3	MB	"3"		
4		"4"	POMODE	
5		"5"	.00	
6	"1"	"6"	$\overline{\text{FCENTS}}$	
7	"1" REG07	"7"	POMODE /FCENTS .00	

FIG.—15.

BIT	BAKSTA 2 HEX	KEYSTA 34 HEX	PRTSTA 35 HEX	(0-3) (38-3B HEX)	STEPTK (0-3) (3C-3F HEX)
0					
1			(0-3)		
2		CC	"0" "1"	(BCD)	(BCD)
3					
4					("1" ")
5		.00			"1"
6		FCENTS	(0-7)		"1" 5 "0"/ "1"
7	PWRLSS	/FCENTS .00	"0" "1"		"1"

FIG.__16.

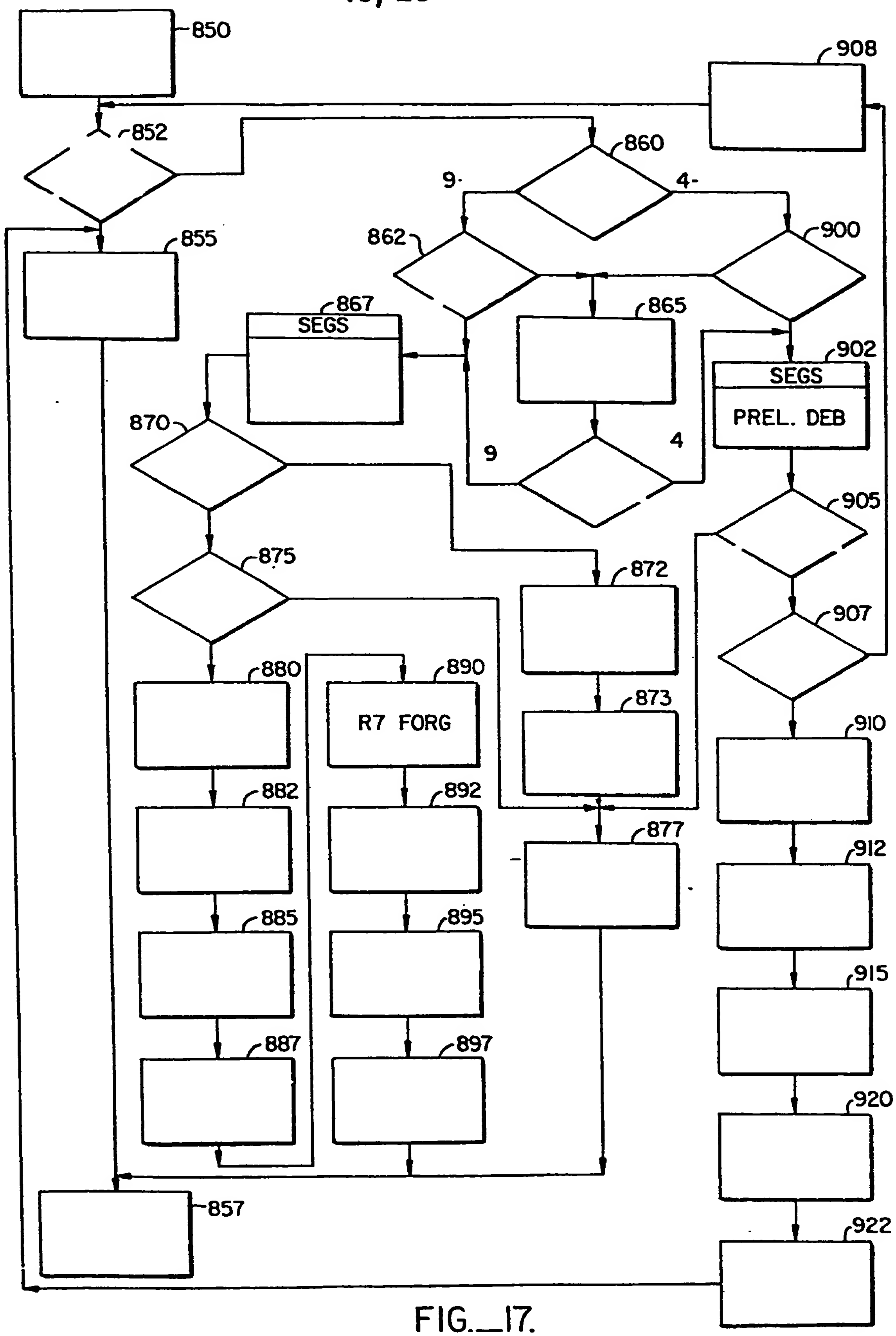
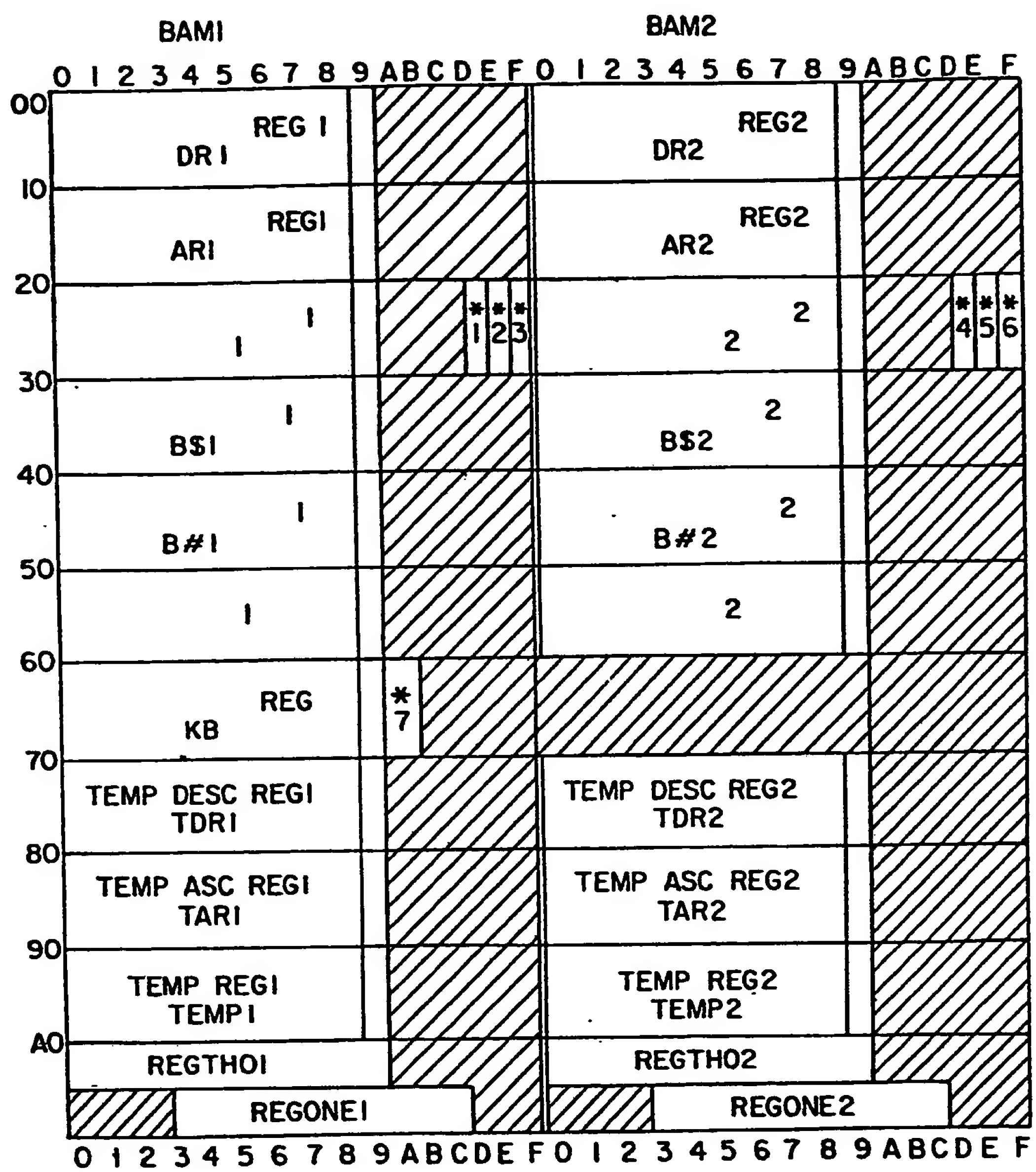

850
908
852
860
9
4
862
900
855
867
SEGS
865
902
SEGS
PREL. DEB
870
9
4
905
875
872
907
880
890
R7 FORG
873
910
882
892
877
912
885
895
915
887
897
920
857
922

FIG._17.



*1 BAMI

*2 MAXPO BAMI

*3 METYP BAMI

*4 BAM2

*5 MAXPO BAM2

*6 METYP BAM2

*7 KBCC

FIG._18.

	TDR1	TDR2	TAR1	TAR2		
CHKTOT	DR	DR	AR	AR	AR+DR	
	DR-KB	DR	AR+KB	AR	(AR+KB) + (DR-KB)	
	DR	DR	AR	AR	AR+DR	
	DR±KB	DR	AR	AR	(DR±KB) + AR	(AR+DR) ± KB

FIG._19.

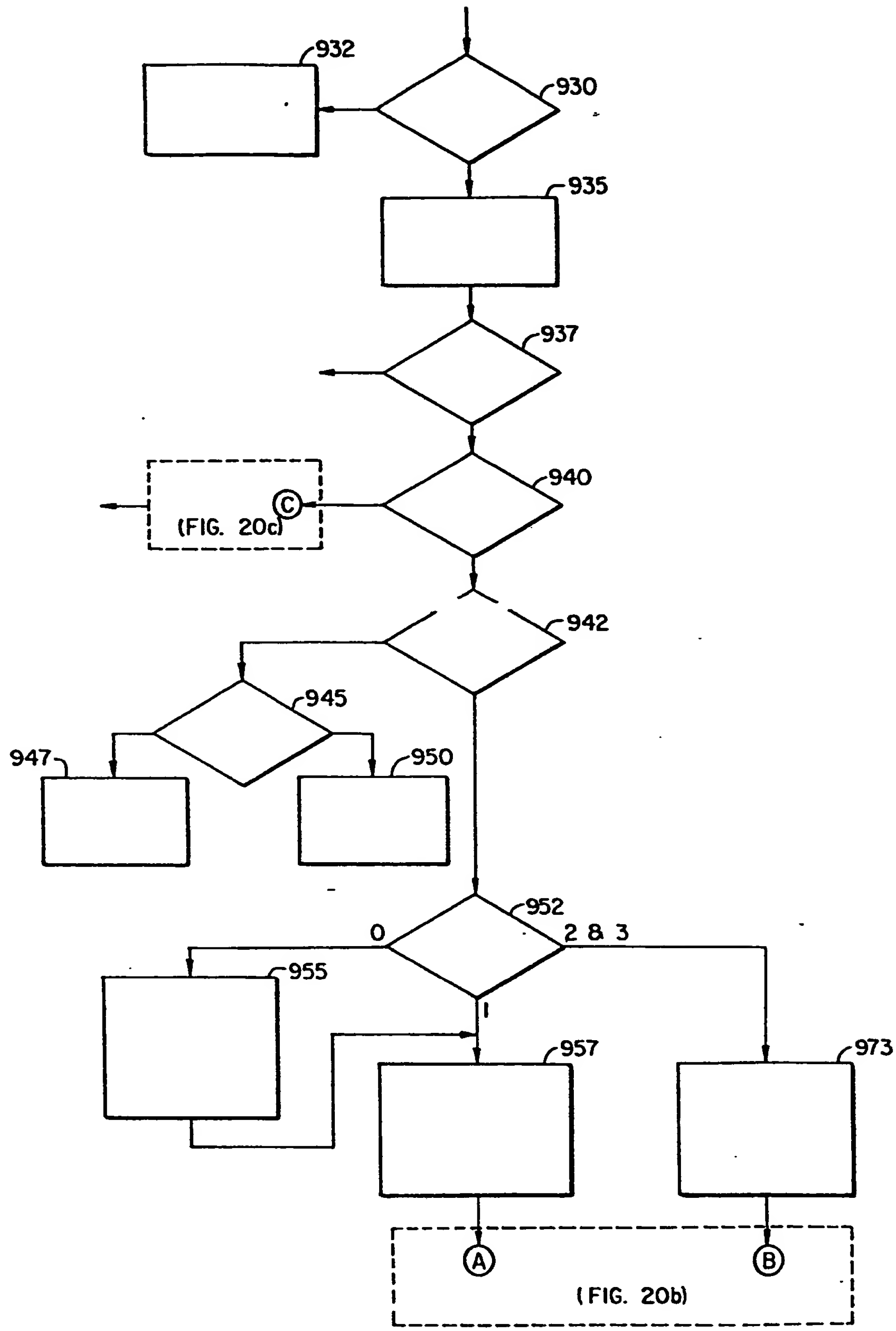

932
930
935
937
940
C
(FIG. 20c)
942
945
947
950
952
0
2 & 3
1
955
957
973
A
B
(FIG. 20b)

FIG._20a.

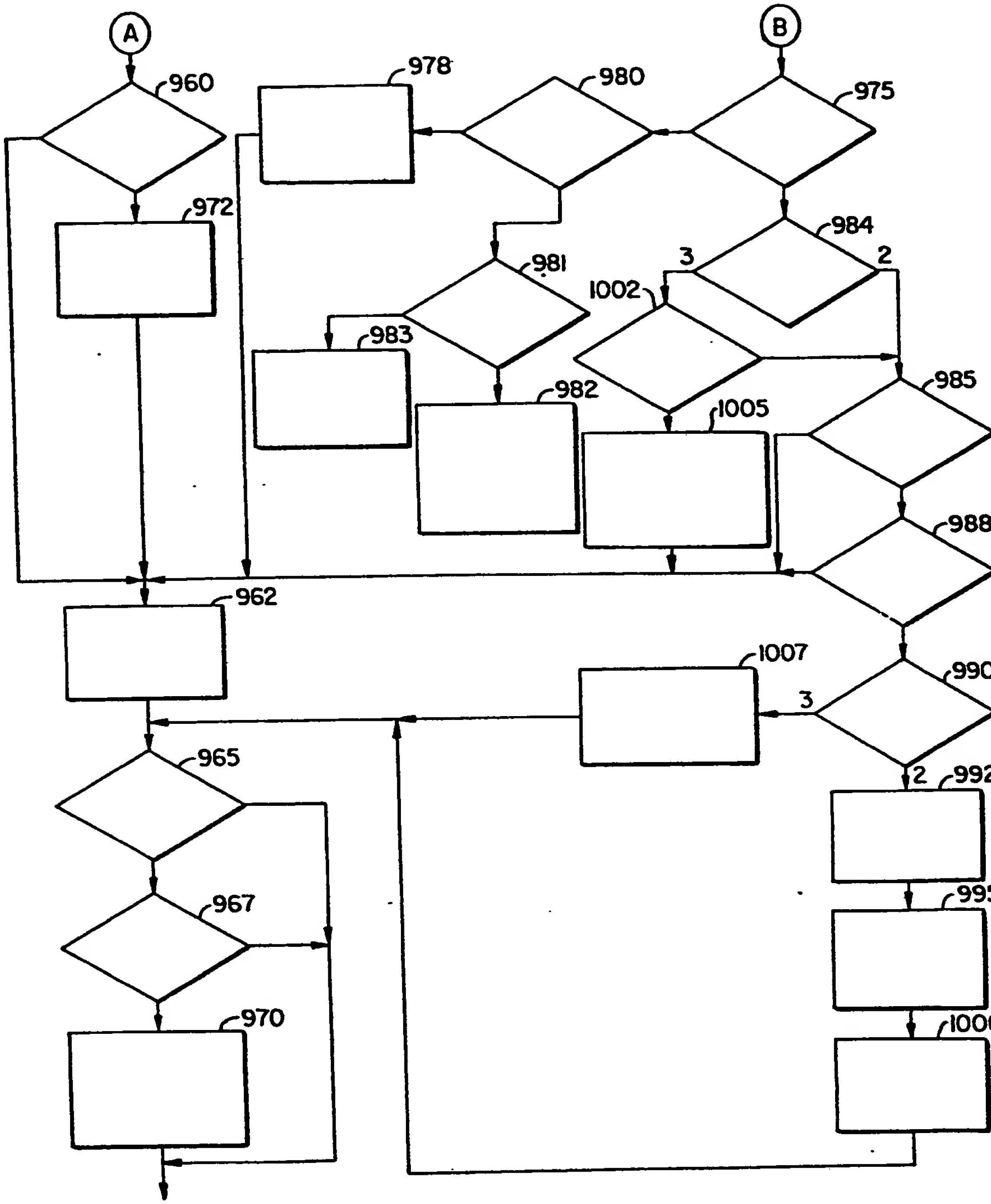

A
B
960
972
978
980
975
981
983
982
984
3
2
1002
1005
985
988
962
1007
990
3
2
992
965
967
995
970
1000

FIG._20b.

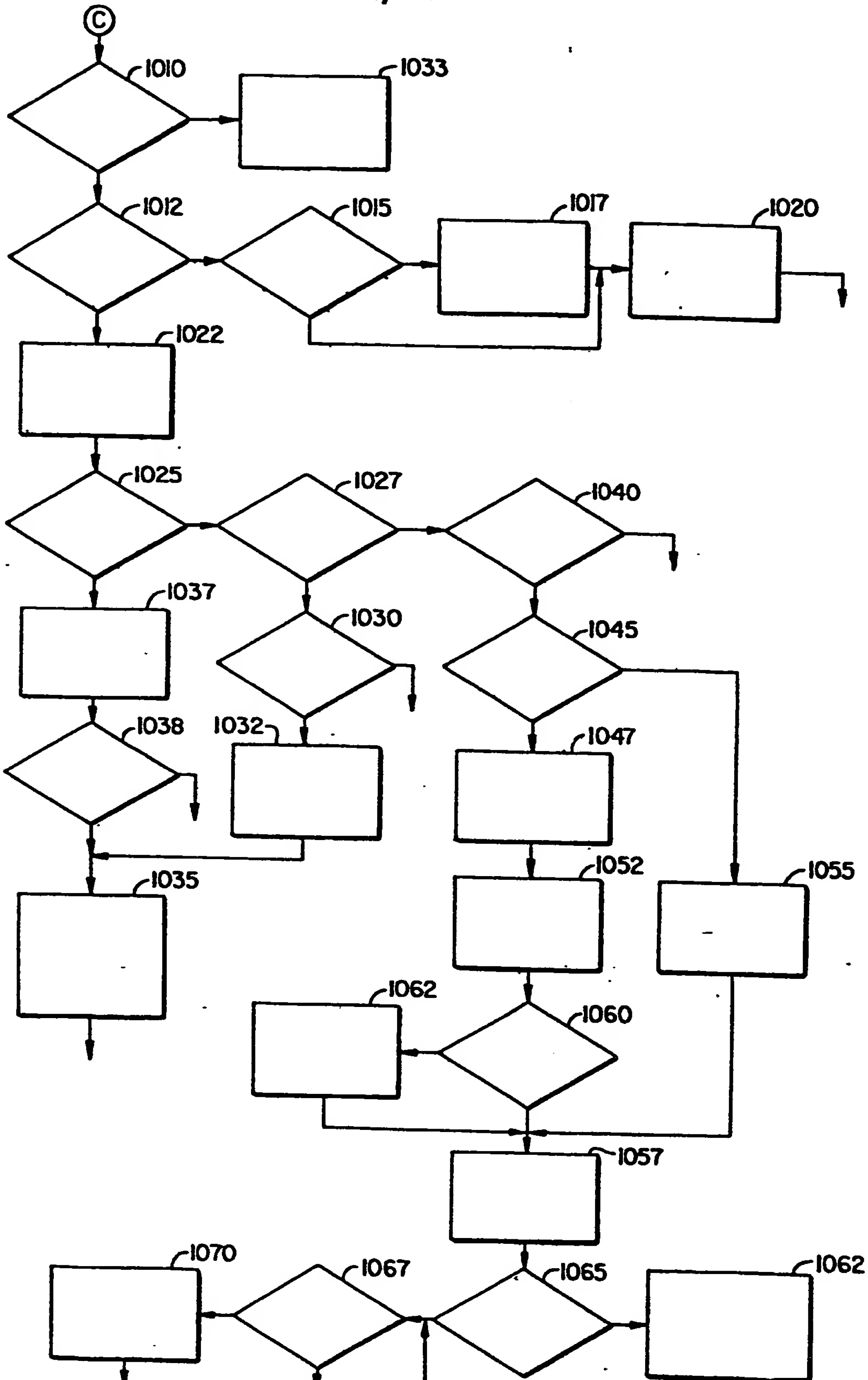

C
1010
1033
1012
1015
1017
1020
1022
1025
1027
1040
1037
1030
1045
1038
1032
1047
1052
1055
1035
1062
1060
1057
1070
1067
1065
1062

FIG._20c.

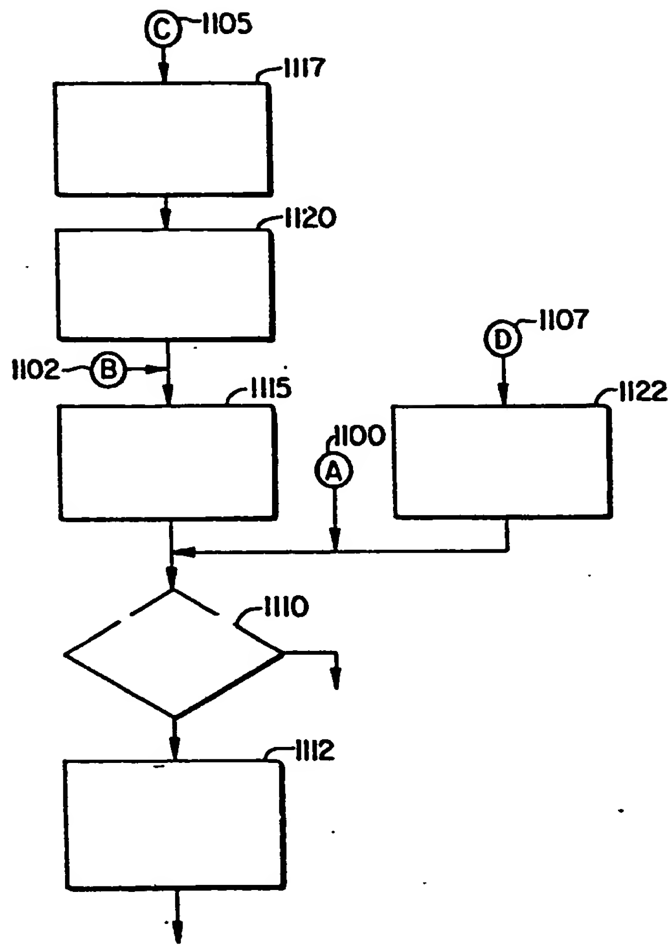


FIG._21a.

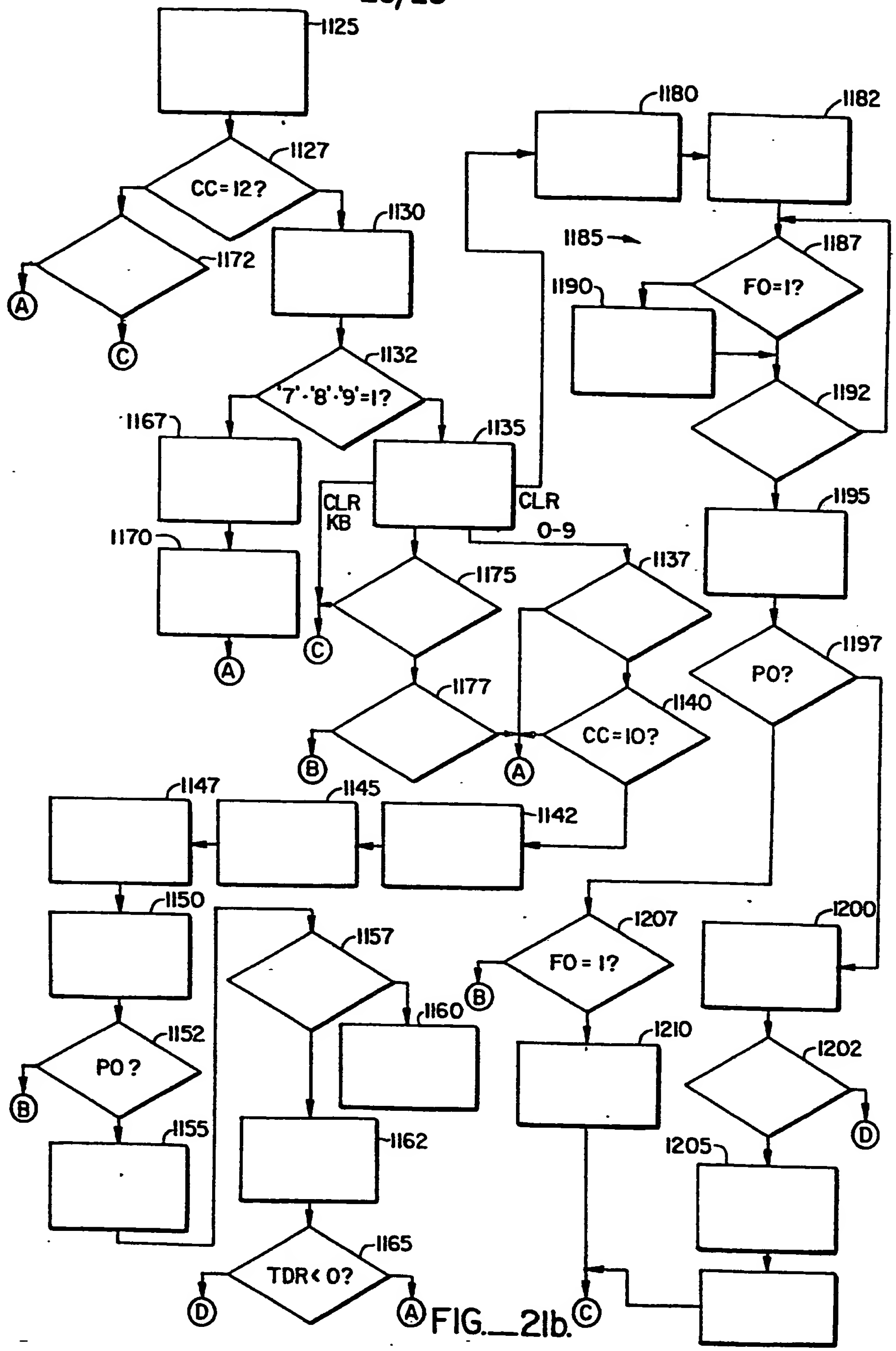

1125
CC=12?
1127
1130
1172
A
C
1132
'7'·'8'·'9'=1?
1167
1135
CLR KB
CLR
0-9
1170
1175
1137
1177
1140
CC=10?
B
1180
1182
1185
1187
F0=1?
1190
1192
1195
1197
P0?
1147
1145
1142
1150
1157
1160
1152
P0 ?
1155
1162
1165
TDR< 0?
D
1207
F0 = 1?
1210
1200
1202
1205

FIG.__21b.